List of Buffer Properties

Input
- Buffer Size
- Onboard Buffer Size

Output
- Buffer Size
- Onboard Buffer Size
List of Calibration Info Properties

Self Calibration
  Is Supported
  Last Self Calibration Temperature

External Calibration
  Recommended Interval
  Last External Calibration Temperature

User-Defined Information
  Information
  Max Size

More
  Device Temperature
List of Channel Properties

Analog Input
- Maximum Value
- Minimum Value
- Custom Scale Name
- Measurement Type

Voltage
- Units
- dB Reference

AC RMS Voltage
- Units

Temperature
- Units

Thermocouple
- Type
- Scale Type
- CJC Source
- CJC Value
- CJC Channel

RTD
- Type
- R0

Custom
- A
- B
- C

Thermistor
- A
- B
- C
- R1

Advanced
- Force Read from Channel

Current
- Units

AC RMS Current
- Units

Strain
- Units

Strain Gage
- Gage Factor
- Poisson Ratio
- Configuration
### Initial Bridge Voltage

### Shunt Cal
- **Shunt Cal Enable**
- **Shunt Cal Select**
- **Gain Adjustment**

### Balance
- **Coarse Potentiometer**
- **Fine Potentiometer**

### Current Shunt Resistor
- **Location**
- **Value**

### Excitation
- **Source**
- **Value**

### Advanced
- **Use Excitation For Scaling**
- **Use Multiplexed Excitation**
- **Actual Excitation Value**
- **DC or AC**
- **Voltage or Current**

### AC Excitation
- **Frequency**
- **Synchronization Enable**
- **Wire Mode**

### Probe/Attenuator
- **Attenuation Value**

### Filter

#### Analog Lowpass
- **Enable**
- **Cutoff Frequency**

#### Advanced

### Switched Capacitor
- **Clock Source**
- **External Clock Frequency**
- **External Clock Divisor**
- **Output Clock Divisor**

### Digitizer/ADC
- **Resolution Units**
- **Resolution Value**
- **Raw Sample Size**
- **Raw Sample Justification**
- **Timing Mode**

### Dither
- **Enable**
Channel Calibration
- Has Valid Calibration Information
- Enable Calibration
- Apply Calibration If Expired

Scaling Parameters
- Scale Type
  - Table
    - Pre-Scaled Values
    - Scaled Values
  - Polynomial
    - Forward Coefficients
    - Reverse Coefficients

Operator Name
- Description

Verification
- Reference Values
- Acquired Values

Advanced
- Range
  - High
  - Low
- Gain
  - Gain Value
- Sample and Hold Enable

High Accuracy Settings
- Auto Zero Mode

Data Transfer and Memory
- Data Transfer Mechanism
- Data Transfer Request Condition
- Data Transfer Custom Threshold
- Memory Mapping for Programmed IO Enable

Compression
- Raw Data Compression Type
- Lossy LSB Removal
  - Compressed Sample Size

Device Scaling Coefficients
- Device Scaling Coefficients
- Enhanced Alias Rejection Enable

Analog Output
- Maximum Value
- Minimum Value
- Custom Scale Name
- Output Type
Voltage
Units
Current Limit

Current
Units

Function Generation
Type
Frequency
Amplitude
Offset

Square
DutyCycle

Modulation
Type
FM Deviation

General Properties
Output Configuration
Output Impedance
Load Impedance
Idle Output Behavior
Terminal Configuration

DAC
Resolution Units
Resolution Value

Range
High
Low

Reference Voltage
Connect DAC Reference to Ground
Allow Connecting DAC Reference to Ground at Runtime
Source
External Source
Value

Offset Voltage
Source
External Source
Value
Reglitching Enable

Advanced
Gain
Gain Value

Data Transfer and Memory
Use Only Onboard Memory
Data Transfer Mechanism
Data Transfer Request Condition
Memory Mapping for Programmed IO Enable

Device Scaling Coefficients
Device Scaling Coefficients
Enhanced Image Rejection Enable

Digital Input
 Invert Lines
 Number of Lines

Digital Filter
 Enable
 Minimum Pulse Width
 Tristate
 Logic Family

General Properties

Advanced

Data Transfer and Memory
 Use Only Onboard Memory
 Data Transfer Mechanism
 Data Transfer Request Condition
 Memory Mapping for Programmed IO Enable

Digital Output
 Output Drive Type
 Invert Lines
 Number of Lines
 Tristate

Line States
 Start State
 Paused State
 Done State
 Logic Family

General Properties

Advanced

Data Transfer and Memory

Counter Input
 Maximum Value
 Minimum Value
 Custom Scale Name
 Measurement Type

Frequency
Measurement Specifications
  Method
  High Frequency
    Measurement Time
  Large Range
    Divisor
Digital Filter
  Enable
  Minimum Pulse Width
Timebase
  Source
  Rate
Digital Synchronization
  Enable
Period
  Units
  Input Terminal
  Starting Edge
Measurement Specifications
  Method
  High Frequency
    Measurement Time
  Large Range
    Divisor
Digital Filter
  Enable
  Minimum Pulse Width
Timebase
  Source
  Rate
Digital Synchronization
  Enable
Count Edges
  Input Terminal
Count Direction
  Direction
  Terminal
Digital Filter
  Enable
Minimum Pulse Width

**Timebase**
- **Source**
- **Rate**

**Digital Synchronization**
- **Enable**
- **Initial Count**
- **Active Edge**

**Digital Filter**
- **Enable**
- **Minimum Pulse Width**

**Timebase**
- **Source**
- **Rate**

**Digital Synchronization**
- **Enable**

**Position**

**Angular Encoder**
- **Units**
- **Pulses Per Revolution**
- **Initial Angle**

**Linear Encoder**
- **Units**
- **Distance Per Pulse**
- **Initial Position**
- **Decoding Type**

**A Input**
- **Terminal**

**Digital Filter**
- **Enable**
- **Minimum Pulse Width**

**Timebase**
- **Source**
- **Rate**

**Digital Synchronization**
- **Enable**

**B Input**
- **Terminal**

**Digital Filter**
- **Enable**
- **Minimum Pulse Width**

**Timebase**
- **Source**
Rate
Digital Synchronization
Enable
Z Input
Terminal
Digital Filter
Enable
Minimum Pulse Width
Timebase
Source
Rate
Digital Synchronization
Enable
Z Index Enable
Z Index Value
Z Index Phase
Pulse Width
Units
Input Terminal
Starting Edge
Digital Filter
Enable
Minimum Pulse Width
Timebase
Source
Rate
Digital Synchronization
Enable
Two Edge Separation
Units
First
Input Terminal
Edge
Digital Filter
Enable
Minimum Pulse Width
Timebase
Source
Rate
Digital Synchronization
Enable
Second
Input Terminal
Digital Filter
  Enable
  Minimum Pulse Width
Timebase
  Source
  Rate
Digital Synchronization
  Enable
Semi-Period
  Units
  Source
  Starting Edge
Digital Filter
  Enable
  Minimum Pulse Width
Timebase
  Source
  Rate
Digital Synchronization
  Enable
Timestamp
  Units
  Source
  Initial Seconds
GPS
  Synchronization Method
  Source
General Properties
  Counter Timebase
    Source
    Rate
    Source
    Active Edge
Digital Filter
  Enable
  Minimum Pulse Width
Timebase
  Source
  Rate
Digital Synchronization
  Enable
More
  Count
  Source
  Output State
### Terminal Count Reached

- Counter Timebase Master Timebase Divisor

### Advanced

#### Data Transfer and Memory
- Data Transfer Mechanism
- Number Of Possibly Invalid Samples
- Duplicate Count Prevention
- Prescaler

### Counter Output

- Output Type

#### Pulse

- Idle State
- Output Terminal

#### Time

- Output Type
- Units
- High Time
- Low Time
- Initial Delay

#### Frequency

- Duty Cycle
- Units
- Frequency
- Initial Delay

#### Ticks

- High Ticks
- Low Ticks
- Initial Delay

### General Properties

#### Counter Timebase

- Source
- Rate
- Active Edge

#### Digital Filter

- Enable
- Minimum Pulse Width

#### Timebase

- Source
- Rate

#### Digital Synchronization

- Enable

### More

- Count
- Output State
- Auto Increment Count
General Properties

- Channel Type
- Physical Channel Name
- Description
- Is Global
List of Device Properties

Device Is Simulated

Identification
  Product Category
  Product Type
  Product Number
  Device Serial Number

Chassis
  Module Device Names
  Analog Triggering Supported
  Digital Triggering Supported

I/O Type

Analog Input
  Physical Channels

Timing
  Maximum Single Channel Rate
  Maximum Multiple Channel Rate
  Minimum Rate
  Simultaneous Sampling Supported

Trigger
  Trigger Usage

Voltage
  Ranges
  Internal Excitation
    Discrete Values
    Range Values

Current
  Ranges
  Internal Excitation
    Discrete Values

Frequency
  Ranges
  Gains
  Couplings

Filter
  Analog Low Pass
    Discrete Values
    Range Values

Cutoff Frequency
  Discrete Values
  Range Values

Analog Output
  Physical Channels
### Timing
- Sample Clock Supported
- Maximum Rate
- Minimum Rate

### Trigger
- Trigger Usage

### Voltage
- Ranges

### Current
- Ranges
- Gains

### Digital Input
- Lines
- Ports
- Timing
  - Maximum Rate
- Trigger
  - Trigger Usage

### Digital Output
- Lines
- Ports
- Timing
  - Maximum Rate
- Trigger
  - Trigger Usage

### Counter Input
- Physical Channels
- Trigger
  - Trigger Usage

### Counter Output
- Physical Channels
- Trigger
  - Trigger Usage

### Location
- Bus Type
- Bus
<table>
<thead>
<tr>
<th>Location</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Bus Number</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Device Number</strong></td>
</tr>
<tr>
<td>PXI</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Chassis Number</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Slot Number</strong></td>
</tr>
<tr>
<td>CompactDAQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Chassis Device Name</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Slot Number</strong></td>
</tr>
</tbody>
</table>

Terminals
List of Export Signal Properties

Clocks

AI Convert Clock
  Output Terminal

Pulse
  Polarity

10MHz Reference Clock
  Output Terminal

20MHz Timebase
  Output Terminal

Sample Clock
  Output Behavior
  Output Terminal
  Delay Offset

Pulse
  Polarity

Sample Clock Timebase
  Output Terminal

Divided Sample Clock Timebase
  Output Terminal

Triggers

Advance Trigger
  Output Terminal

Pulse
  Polarity
  Width Units
  Width Value

Pause Trigger
  Output Terminal

Level
  Active Level

Reference Trigger
  Output Terminal

Pulse
  Polarity

Start Trigger
  Output Terminal

Pulse
  Polarity
Events

**Advance Complete Event**
- Output Terminal
- Delay Value

**Pulse**
- Polarity
- Width Value

**AI Hold Complete Event**
- Output Terminal

**Pulse**
- Polarity

**Change Detection Event**
- Output Terminal

**Pulse**
- Polarity

**Counter Output Event**
- Output Terminal
- Output Behavior

**Pulse**
- Polarity

**Toggle**
- Idle State

**Handshake Event**
- Output Terminal
- Output Behavior
- Delay Value

**Interlocked**
- Asserted Level
- Assert on Start
- Deassert Delay Value

**Pulse**
- Polarity
- Width Value

**Ready For Transfer Event**
- Output Terminal

**Level**
- Active Level
- Deassert Condition
- Deassert Condition Custom Threshold

**Data Active Event**
- Output Terminal

**Level**
Active Level

Ready For Start Event

Level

Synchronization Pulse Event

Watchdog Timer Expired Event

Output Terminal
## List of Persisted Channel Properties

<table>
<thead>
<tr>
<th>Author</th>
<th>Allow Interactive Editing?</th>
<th>Allow Interactive Deletion?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Allow Interactive Editing?</td>
<td>Allow Interactive Deletion?</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
List of Persisted Task Properties

- Author
- Allow Interactive Editing?
- Allow Interactive Deletion?
List of Physical Channel Properties

Analog Input
  Input Configuration
    Terminal Configurations

Analog Output
  Output Configuration
    Terminal Configurations

Advanced
  Manual Control
    Enable
    Amplitude
    Frequency

Digital Input
  Port Width
  Timing
    Sample Clock Supported
    Change Detection Supported

Digital Output
  Port Width
  Timing
    Sample Clock Supported

TEDS
  ManufacturerID
  Model Number
  Serial Number
  Version Number
  Version Letter
  BitStream
  TemplateIDs
List of Read Properties

-RelativeTo
-Offset
-Channels to Read
-Read All Available Samples
-Auto Start
-OverWrite Mode

Status

-Current Read Position
-Available Samples Per Channel
-Total Samples Per Channel Acquired

Overcurrent

-Overcurrent Channels Exist
-Overcurrent Channels

Open Current Loop

-Open Current Loop Channels Exist
-Open Current Loop Channels

Overload

-Overloaded Channels Exist
-Overloaded Channels

Advanced

-ChangeDetection

Advanced

-Raw Data Width

Digital Input

-Number of Channels

Wait Mode

-Sleep Time
List of Real-Time Properties

- Convert Late Errors To Warnings
- Number Of Warmup Iterations
- Wait For Next Sample Clock Wait Mode
- Report Missed Samples
- Write Recovery Mode
### List of Scale Properties

<table>
<thead>
<tr>
<th>Description</th>
<th>Scale Type</th>
<th>Linear</th>
<th>Map</th>
<th>Polynomial</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slope</td>
<td></td>
<td>Scaled Maximum Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y-Intercept</td>
<td></td>
<td>Pre-Scaled Maximum Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scaled Minimum Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pre-Scaled Minimum Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Forward Coefficients</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reverse Coefficients</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Scaled Values</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pre-Scaled Values</td>
<td></td>
</tr>
</tbody>
</table>
List of Switch Channel Properties

Usage

Capability

Max AC Carry Current
Max AC Switching Current
Max AC Carry Power
Max AC Switching Power
Max DC Carry Current
Max DC Switching Current
Max DC Carry Power
Max DC Switching Power
Max AC Voltage
Max DC Voltage
Wire Mode
Bandwidth
Impedance
List of Switch Device Properties

Settling Time
Auto Connect Analog Bus
Power Down Latching Relays After Settling Is Settled

Capability
   Relay List
   Number of Relays
   Switch Channel List
   Number of Switch Channels
   Number of Rows
   Number of Columns
   Topology
List of Switch Scan Properties

- Break Mode
- Repeat Mode
- Is Waiting For Advance
List of System Properties

Global Channels
Scales
Tasks
Device Names

System

NI-DAQ Version

Major
Minor
<table>
<thead>
<tr>
<th>List of Task Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Channels</strong></td>
</tr>
<tr>
<td><strong>Number of Channels</strong></td>
</tr>
<tr>
<td><strong>Devices</strong></td>
</tr>
<tr>
<td><strong>Number of Devices</strong></td>
</tr>
<tr>
<td><strong>Task Done</strong></td>
</tr>
</tbody>
</table>
List of Timing Properties

Sample Quantity
- Sample Mode
- Samples Per Channel

Sample Timing Type
- Sample Clock
  - Rate
  - Maximum Rate
  - Source
  - Active Edge
  - Underflow Behavior
  - Timebase Divisor

Timebase
- Rate
- Source
- Active Edge
- Master Timebase Divisor

Digital Filter
- Enable
- Minimum Pulse Width

Timebase
- Source
- Rate

Digital Synchronization
- Enable

Handshake
- Delay After Transfer
- Start Condition
- Sample Input Data When

Change Detection

Digital Input
- Rising Edge Physical Channels
- Falling Edge Physical Channels

On Demand
- Simultaneous Analog Output Enable

More

AI Convert
- Rate
- Maximum Rate
- Source
- Active Edge
- Timebase Divisor

Timebase
Source

Delay From Sample Clock
  Delay Units
  Delay

Master Timebase
  Rate
  Source

Reference Clock
  Rate
  Source

Synchronization Pulse
  Source
  Synchronization Time
  Minimum Delay To Start

Advanced
  Sample Timing Engine
List of Trigger Properties

Start

  Trigger Type

Digital Edge

  Source
  Edge

Digital Filter

  Enable
  Minimum Pulse Width

Timebase

  Source
  Rate

Digital Synchronization

  Enable

Digital Pattern

  Source
  Pattern
  Trigger When

Analog Edge

  Source
  Slope
  Level
  Hysteresis
  Coupling

Analog Window

  Source
  Trigger When
  Top
  Bottom
  Coupling

More

  Delay
  Delay Units
  Retriggerable

Reference

  Trigger Type
  Pretrigger Samples per Channel

Digital Edge

  Source
  Edge

Digital Pattern

  Source
  Pattern
  Trigger When
Analog Edge
  Source
  Slope
  Level
  Hysteresis
  Coupling

Analog Window
  Source
  Trigger When
  Top
  Bottom
  Coupling

More

Advance
  Trigger Type

Digital Edge
  Source
  Edge

Digital Filter
  Enable

Handshake
  Trigger Type

Interlocked
  Source
  Asserted Level

Pause
  Trigger Type

Analog Level
  Source
  Pause When
  Level
  Hysteresis
  Coupling

Analog Window
  Source
  Pause When
  Top
  Bottom
  Coupling

Digital Level
  Source
  Pause When

Digital Filter
  Enable
<table>
<thead>
<tr>
<th>Minimum Pulse Width</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timebase</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td></td>
</tr>
<tr>
<td><strong>Digital Synchronization</strong></td>
<td></td>
</tr>
<tr>
<td>Enable</td>
<td></td>
</tr>
<tr>
<td><strong>Digital Pattern</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Pattern</td>
<td></td>
</tr>
<tr>
<td>Pause When</td>
<td></td>
</tr>
<tr>
<td><strong>Arm Start</strong></td>
<td></td>
</tr>
<tr>
<td>Trigger Type</td>
<td></td>
</tr>
<tr>
<td><strong>Digital Edge</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Edge</td>
<td></td>
</tr>
<tr>
<td><strong>Digital Filter</strong></td>
<td></td>
</tr>
<tr>
<td>Enable</td>
<td></td>
</tr>
<tr>
<td>Minimum Pulse Width</td>
<td></td>
</tr>
<tr>
<td><strong>Timebase</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td></td>
</tr>
<tr>
<td><strong>Digital Synchronization</strong></td>
<td></td>
</tr>
<tr>
<td>Enable</td>
<td></td>
</tr>
</tbody>
</table>
List of Watchdog Properties

Timeout

Expiration Trigger
  Trigger Type
    Digital Edge
      Source
      Edge

Expiration States

Digital Output
  Expiration State

Status
  Expired
# List of Write Properties

**Relative To**
- Offset
- Regeneration Mode

## Status

- **Current Write Position**
- **Overcurrent**
  - Overcurrent Channels Exist
  - Overcurrent Channels
- **Open Current Loop**
  - Open Current Loop Channels Exist
  - Open Current Loop Channels
- **Power Supply Fault**
  - Power Supply Fault Channels Exist
  - Power Supply Fault Channels
- **Space Available in Buffer**
- **Total Samples Per Channel Generated**

## Advanced

- **Raw Data Width**
- **Number of Channels**
- **Wait Mode**
- **Sleep Time**
- **Next Write Is Last**

## Digital Output

- **Number of Booleans Per Channel**
**Input >> Buffer Size**

**Data Type:**  uInt32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetBufInputBufSize`
- `DAQmxSetBufInputBufSize`
- `DAQmxResetBufInputBufSize`
**Input >> Onboard Buffer Size**

**Data Type:** uInt32

**Description:** Indicates in samples per channel the size of the onboard input buffer of the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetBufInputOnbrdBufSize](#)
Output >> Buffer Size

**Data Type:** uInt32

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetBufOutputBufSize
- DAQmxSetBufOutputBufSize
- DAQmxResetBufOutputBufSize
**Output >> Onboard Buffer Size**

**Data Type:** uInt32

**Description:** Specifies in samples per channel the size of the onboard output buffer of the device.

You can get/set/reset this property using:

DAQmxGetBufOutputOnbrdBufSize
DAQmxSetBufOutputOnbrdBufSize
DAQmxResetBufOutputOnbrdBufSize
Self Calibration >> Is Supported

Data Type: bool32
Description: Indicates whether the device supports self calibration.
Restrictions: Not Settable

You can get this property using:

DAQmxGetSelfCalSupported
Self Calibration >> Last Self Calibration Temperature

**Data Type:** float64

**Description:** Indicates in degrees Celsius the temperature of the device at the time of the last self calibration. Compare this temperature to the current onboard temperature to determine if you should perform another calibration.

The temperature returned by this property is the calibration temperature as measured by an onboard temperature sensor and may differ from the temperature displayed on a printed calibration certificate. Calibration certificates usually display the ambient temperature rather than the onboard temperature.

Using Traditional NI-DAQ (Legacy) to perform calibration does not update this property. This property is updated only when you use NI-DAQmx to perform calibration.

**Restrictions:** Not Settable

You can get this property using:

*DAQmxGetSelfCalLastTemp*
External Calibration >> Recommended Interval

**Data Type:**  uInt32

**Description:**  Indicates in months the National Instruments recommended interval between each external calibration of the device.

**Restrictions:**  Not Settable

You can get this property using:

DAQmxGetExtCalRecommendedInterval
**External Calibration >> Last External Calibration Temperature**

**Data Type:** float64

**Description:** Indicates in degrees Celsius the temperature of the device at the time of the last external calibration. Compare this temperature to the current onboard temperature to determine if you should perform another calibration.

The temperature returned by this property is the calibration temperature as measured by an onboard temperature sensor and may differ from the temperature displayed on a printed calibration certificate. Calibration certificates usually display the ambient temperature rather than the onboard temperature.

Using Traditional NI-DAQ (Legacy) to perform calibration does not update this property. This property is updated only when you use NI-DAQmx to perform calibration.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetExtCalLastTemp`
**User-Defined Information >> Information**

**Data Type:** char*

**Description:** Specifies a string that contains arbitrary, user-defined information. This number of characters in this string can be no more than Max Size.

You can get/set this property using:

- `DAQmxGetCalUserDefinedInfo`
- `DAQmxSetCalUserDefinedInfo`
User-Defined Information >> Max Size

**Data Type:** uint32

**Description:** Indicates the maximum length in characters of Information.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetCalUserDefinedInfoMaxSize`
More >> Device Temperature

**Data Type:** float64

**Description:** Indicates in degrees Celsius the current temperature of the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetCalDevTemp`
**Analog Input >> Maximum Value**

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAIMax`
- `DAQmxSetAIMax`
- `DAQmxResetAIMax`
Analog Input >> Minimum Value

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAIMin
- DAQmxSetAIMin
- DAQmxResetAIMin
**Analog Input >> Custom Scale Name**

**Data Type:** char*

**Description:** Specifies the name of a custom scale for the channel.

You can get/set/reset this property using:

- DAQmxGetAICustomScaleName
- DAQmxSetAICustomScaleName
- DAQmxResetAICustomScaleName
**Analog Input >> Measurement Type**

**Data Type:** int32

**Description:** Indicates the measurement to take with the analog input channel and in some cases, such as for temperature measurements, the sensor to use.

**Restrictions:** Not Settable

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Voltage</td>
<td>10322</td>
<td>Voltage measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_VoltageRMS</td>
<td>10350</td>
<td>Voltage RMS measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_Current</td>
<td>10134</td>
<td>Current measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_CurrentRMS</td>
<td>10351</td>
<td>Current RMS measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_Voltage_CustomWithExcitation</td>
<td>10323</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_Freq_Voltage</td>
<td>10181</td>
<td>Frequency measurement using a frequency to voltage converter.</td>
</tr>
<tr>
<td>DAQmx_Val_Resistance</td>
<td>10278</td>
<td>Resistance measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_Temp_TC</td>
<td>10303</td>
<td>Temperature measurement using a thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_Temp_Thrmstr</td>
<td>10302</td>
<td>Temperature measurement using a thermistor.</td>
</tr>
<tr>
<td>DAQmx_Val_Temp_RTD</td>
<td>10301</td>
<td>Temperature measurement using an RTD.</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DAQmx_Val_Temp_BuiltInSensor</td>
<td>10311</td>
<td>Temperature measurement using a built-in sensor on a terminal block or device. On SCXI modules, for example, this could be the CJC sensor.</td>
</tr>
<tr>
<td>DAQmx_Val_Strain_Gage</td>
<td>10300</td>
<td>Strain measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_Position_LVDT</td>
<td>10352</td>
<td>Position measurement using an LVDT.</td>
</tr>
<tr>
<td>DAQmx_Val_Position_RVDT</td>
<td>10353</td>
<td>Position measurement using an RVDT.</td>
</tr>
<tr>
<td>DAQmx_Val_Accelerometer</td>
<td>10356</td>
<td>Acceleration measurement using an accelerometer.</td>
</tr>
<tr>
<td>DAQmx_Val_SoundPressure_Microphone</td>
<td>10354</td>
<td>Sound pressure measurement using a microphone.</td>
</tr>
<tr>
<td>DAQmx_Val_TEDS_Sensor</td>
<td>12531</td>
<td>Measurement type defined by TEDS.</td>
</tr>
</tbody>
</table>

You can get this property using:

[DAQmxGetAI_measType](https://docs.ni.com/display/EN/daqmx)
## Analog Input >> Voltage >> Units

**Data Type:** int32  
**Description:** Specifies the units to use to return voltage measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Volts</td>
<td>10348</td>
<td>Volts.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a <a href="#">custom scale</a> specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAIVoltageUnits](#)
- [DAQmxSetAIVoltageUnits](#)
- [DAQmxResetAIVoltageUnits](#)
Analog Input >> Voltage >> dB Reference

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAIVoltagedBRef
- DAQmxSetAIVoltagedBRef
- DAQmxResetAIVoltagedBRef
Analog Input >> Voltage >> AC RMS Voltage >> Units

**Data Type:** int32

**Description:** Specifies the units to use to return voltage RMS measurements from the channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Volts</th>
<th>10348</th>
<th>Volts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIVoltageACRMSUnits
- DAQmxSetAIVoltageACRMSUnits
- DAQmxResetAIVoltageACRMSUnits
Analog Input >> Temperature >> Units

**Data Type:** int32

**Description:** Specifies the units to use to return temperature measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DegC</td>
<td>10143</td>
<td>Degrees Celsius.</td>
</tr>
<tr>
<td>DAQmx_Val_DegF</td>
<td>10144</td>
<td>Degrees Fahrenheit.</td>
</tr>
<tr>
<td>DAQmx_Val_Kelvins</td>
<td>10325</td>
<td>Kelvins.</td>
</tr>
<tr>
<td>DAQmx_Val_DegR</td>
<td>10145</td>
<td>Degrees Rankine.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAITempUnits`
- `DAQmxSetAITempUnits`
- `DAQmxResetAITempUnits`
**Analog Input >> Temperature >> Thermocouple >> Type**

**Data Type:** int32

**Description:** Specifies the type of thermocouple connected to the channel. Thermocouple types differ in composition and measurement range.

### Valid values

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_J_Type_TC</td>
<td>10072</td>
<td>J-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_K_Type_TC</td>
<td>10073</td>
<td>K-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_N_Type_TC</td>
<td>10077</td>
<td>N-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_R_Type_TC</td>
<td>10082</td>
<td>R-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_S_Type_TC</td>
<td>10085</td>
<td>S-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_T_Type_TC</td>
<td>10086</td>
<td>T-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_B_Type_TC</td>
<td>10047</td>
<td>B-type thermocouple.</td>
</tr>
<tr>
<td>DAQmx_Val_E_Type_TC</td>
<td>10055</td>
<td>E-type thermocouple.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAIThmcplType`
- `DAQmxSetAIThmcplType`
- `DAQmxResetAIThmcplType`
Analog Input >> Temperature >> Thermocouple >> Scale Type

**Data Type:** int32

**Description:** Specifies the method or equation form that the thermocouple scale uses.

### Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Polynomial</td>
<td>10449</td>
<td>Scale values by using an Nth order polynomial equation.</td>
</tr>
<tr>
<td>DAQmx_Val_Table</td>
<td>10450</td>
<td>Map an array of prescaled values to an array of corresponding scaled values, with all other values scaled proportionally.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIThrmcplScaleType
- DAQmxSetAIThrmcplScaleType
- DAQmxResetAIThrmcplScaleType
Analog Input >> Temperature >> Thermocouple >>
CJC Source

Data Type: int32
Description: Indicates the source of cold-junction compensation.
Restrictions: Not Settable

Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_BuiltIn</td>
<td>10200</td>
<td>Use a cold-junction compensation channel built into the terminal block.</td>
</tr>
<tr>
<td>DAQmx_Val_ConstVal</td>
<td>10116</td>
<td>You must specify the cold-junction temperature.</td>
</tr>
<tr>
<td>DAQmx_Val_Chan</td>
<td>10113</td>
<td>Use a channel for cold-junction compensation.</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetAIThrmcplCJCSrc
Analog Input >> Temperature >> Thermocouple >> CJC Value

**Data Type:** float64

**Description:** Specifies the temperature of the cold junction if **CJC Source** is DAQmx_Val_ConstVal. Specify this value in the units of the measurement.

You can get/set/reset this property using:

- DAQmxGetAIThrmcplCJCVal
- DAQmxSetAIThrmcplCJCVal
- DAQmxResetAIThrmcplCJCVal
Analog Input >> Temperature >> Thermocouple >> CJC Channel

**Data Type:** char*

**Description:** Indicates the channel that acquires the temperature of the cold junction if CJC Source is DAQmx_Val_Chan. 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.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetAIThrmcplCJCChan`
**Analog Input >> Temperature >> RTD >> Type**

**Data Type:** int32

**Description:** Specifies the type of RTD connected to the channel.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Pt3750</td>
<td>12481</td>
<td>Pt3750.</td>
</tr>
<tr>
<td>DAQmx_Val_Pt3851</td>
<td>10071</td>
<td>Pt3851.</td>
</tr>
<tr>
<td>DAQmx_Val_Pt3911</td>
<td>12482</td>
<td>Pt3911.</td>
</tr>
<tr>
<td>DAQmx_Val_Pt3916</td>
<td>10069</td>
<td>Pt3916.</td>
</tr>
<tr>
<td>DAQmx_Val_Pt3920</td>
<td>10053</td>
<td>Pt3920.</td>
</tr>
<tr>
<td>DAQmx_Val_Pt3928</td>
<td>12483</td>
<td>Pt3928.</td>
</tr>
<tr>
<td>DAQmx_Val_Custom</td>
<td>10137</td>
<td>Pt3911.</td>
</tr>
</tbody>
</table>

You must use A, B, and C to supply the coefficients for the Callendar-Van Dusen equation.

You can get/set/reset this property using:

- `DAQmxGetAIRTDType`
- `DAQmxSetAIRTDType`
- `DAQmxResetAIRTDType`
Analog Input >> Temperature >> RTD >> R0

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAIRTDR0
- DAQmxSetAIRTDR0
- DAQmxResetAIRTDR0
**Analogue Input >> Temperature >> RTD >> Custom >> A**

**Data Type:** float64

**Description:** Specifies the 'A' constant of the Callendar-Van Dusen equation. NI-DAQmx requires this value when you use a custom RTD.

You can get/set/reset this property using:

- `DAQmxGetAIRTDA`
- `DAQmxSetAIRTDA`
- `DAQmxResetAIRTDA`
Analog Input >> Temperature >> RTD >> Custom >> B

**Data Type:** float64

**Description:** Specifies the 'B' constant of the Callendar-Van Dusen equation. NI-DAQmx requires this value when you use a custom RTD.

You can get/set/reset this property using:

- DAQmxGetAIRTD
- DAQmxSetAIRTD
- DAQmxResetAIRTD
Analog Input >> Temperature >> RTD >> Custom >> C

**Data Type:** float64

**Description:** Specifies the 'C' constant of the Calendar-Van Dusen equation. NI-DAQmx requires this value when you use a custom RTD.

You can get/set/reset this property using:

`DAQmxGetAIRTDC`

`DAQmxSetAIRTDC`

`DAQmxResetAIRTDC`
**Analog Input >> Temperature >> Thermistor >> A**

**Data Type:** float64  
**Description:** Specifies the ‘A’ constant of the Steinhart-Hart thermistor equation.

You can get/set/reset this property using:

- `DAQmxGetAIThrmstrA`
- `DAQmxSetAIThmstrA`
- `DAQmxResetAIThmstrA`
Analog Input >> Temperature >> Thermistor >> B

**Data Type:** float64

**Description:** Specifies the ‘B’ constant of the Steinhart-Hart thermistor equation.

You can get/set/reset this property using:

- `DAQmxGetAIThmstrB`
- `DAQmxSetAIThmstrB`
- `DAQmxResetAIThmstrB`
Analog Input >> Temperature >> Thermistor >> C

**Data Type:** float64

**Description:** Specifies the 'C' constant of the Steinhart-Hart thermistor equation.

You can get/set/reset this property using:

- `DAQmxGetAIThrmstrC`
- `DAQmxSetAIThrmstrC`
- `DAQmxResetAIThrmstrC`
Analog Input >> Temperature >> Thermistor >> R1

**Data Type:** float64

**Description:** Specifies in ohms the value of the reference resistor if you use voltage excitation. NI-DAQmx ignores this value for current excitation.

You can get/set/reset this property using:

- `DAQmxGetAIThmstrR1`
- `DAQmxSetAIThmstrR1`
- `DAQmxResetAIThmstrR1`
**Analog Input >> Temperature >> Advanced >> Force Read from Channel**

**Data Type:** bool32

**Description:** Specifies whether to read from the channel if it is a cold-junction compensation channel. By default, an NI-DAQmx Read function 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.

You can get/set/reset this property using:

- DAQmxGetAIForceReadFromChan
- DAQmxSetAIForceReadFromChan
- DAQmxResetAIForceReadFromChan
**Analog Input >> Current >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return current measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Amps</th>
<th>10342</th>
<th>Amperes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a [custom scale] specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAICurrentUnits](#)
- [DAQmxSetAICurrentUnits](#)
- [DAQmxResetAICurrentUnits](#)
Analog Input >> Current >> AC RMS Current >> Units

**Data Type:** int32

**Description:** Specifies the units to use to return current RMS measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Amps</td>
<td>10342</td>
<td>Amperes.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAICurrentACRMSUnits`
- `DAQmxSetAICurrentACRMSUnits`
- `DAQmxResetAICurrentACRMSUnits`
**Analog Input >> Strain >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return strain measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Strain</th>
<th>10299</th>
<th>Strain.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a <a href="https://example.com/custom-scale">custom scale</a> specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAIStrainUnits](https://example.com/daqmxgetaistrainunits)
- [DAQmxSetAIStrainUnits](https://example.com/daqmxsetaistrainunits)
- [DAQmxResetAIStrainUnits](https://example.com/daqmxresetaistrainunits)
Analog Input >> Strain >> Strain Gage >> Gage Factor

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAlStrainGageGageFactor
- DAQmxSetAlStrainGageGageFactor
- DAQmxResetAlStrainGageGageFactor
Analog Input >> Strain >> Strain Gage >> Poisson Ratio

**Data Type:** float64

**Description:** Specifies the ratio of lateral strain to axial strain in the material you are measuring.

You can get/set/reset this property using:

- `DAQmxGetAIStrainGagePoissonRatio`
- `DAQmxSetAIStrainGagePoissonRatio`
- `DAQmxResetAIStrainGagePoissonRatio`
**Analog Input >> Strain >> Strain Gage >> Configuration**

**Data Type:** int32  
**Description:** Specifies the bridge configuration of the strain gages.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FullBridgeI</td>
<td>Four active gages with two pairs subjected to equal and opposite strains.</td>
</tr>
<tr>
<td>DAQmx_Val_FullBridgeII</td>
<td>Four active gages with two aligned with maximum principal strain and two Poisson gages in adjacent arms.</td>
</tr>
<tr>
<td>DAQmx_Val_FullBridgeIII</td>
<td>Four active gages with two aligned with maximum principal strain and two Poisson gages in opposite arms.</td>
</tr>
<tr>
<td>DAQmx_Val_HalfBridgeI</td>
<td>Two active gages with one aligned with maximum principal strain and one Poisson gage.</td>
</tr>
<tr>
<td>DAQmx_Val_HalfBridgeII</td>
<td>Two active gages with equal and opposite strains.</td>
</tr>
<tr>
<td>DAQmx_Val_QuarterBridgeI</td>
<td>Single active gage.</td>
</tr>
<tr>
<td>DAQmx_Val_QuarterBridgeII</td>
<td>Single active gage and one dummy gage.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIStrainGageCfg
- DAQmxSetAIStrainGageCfg
- DAQmxResetAIStrainGageCfg
**Analog Input >> Resistance >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return resistance measurements.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Ohms</td>
<td>10384</td>
<td>Ohms.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAIResistanceUnits](#)
- [DAQmxSetAIResistanceUnits](#)
- [DAQmxResetAIResistanceUnits](#)
**Analog Input >> Frequency >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return frequency measurements from the channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Hz</th>
<th>10373</th>
<th>Hertz.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

*DAQmxGetAIFreqUnits*
*DAQmxSetAIFreqUnits*
*DAQmxResetAIFreqUnits*
### Analog Input >> Frequency >> Voltage >> Threshold Level

**Data Type:** float64  
**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAIFreqThreshVoltage
- DAQmxSetAIFreqThreshVoltage
- DAQmxResetAIFreqThreshVoltage
Analog Input >> Frequency >> Voltage >> Hysteresis

**Data Type:** float64

**Description:** Specifies in volts a window below Threshold Level. The input voltage must pass below Threshold Level minus this value before NI-DAQmx recognizes a waveform repetition at Threshold Level. Hysteresis can improve the measurement accuracy when the signal contains noise or jitter.

You can get/set/reset this property using:

- DAQmxGetAIFreqHyst
- DAQmxSetAIFreqHyst
- DAQmxResetAIFreqHyst
**Analog Input >> Position >> LVDT >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return linear position measurements from the channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Meters</th>
<th>10219</th>
<th>Meters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Inches</td>
<td>10379</td>
<td>Inches.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a <a href="#">custom scale</a> specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAILVDTUnits
- DAQmxSetAILVDTUnits
- DAQmxResetAILVDTUnits
Analog Input >> Position >> LVDT >> Sensitivity

**Data Type:** float64  
**Description:** Specifies the sensitivity of the LVDT. This value is in the units you specify with Sensitivity Units. Refer to the sensor documentation to determine this value.

You can get/set/reset this property using:

- DAQmxGetAILVDTSensitivity
- DAQmxSetAILVDTSensitivity
- DAQmxResetAILVDTSensitivity
**Analog Input >> Position >> LVDT >> Sensitivity Units**

**Data Type:** int32

**Description:** Specifies the units of Sensitivity.

<table>
<thead>
<tr>
<th>Valid values</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_mVoltsPerVoltPerMillimeter</td>
<td>12506</td>
<td>mVolts/Volt/mMeter</td>
</tr>
<tr>
<td>DAQmx_Val_mVoltsPerVoltPerMilliInch</td>
<td>12505</td>
<td>mVolts/Volt/0.001 Inch</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAILVDTSensitivityUnits
- DAQmxSetAILVDTSensitivityUnits
- DAQmxResetAILVDTSensitivityUnits
Analog Input >> Position >> RVDT >> Units

**Data Type:** int32

**Description:** Specifies the units to use to return angular position measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Degrees</th>
<th>10146</th>
<th>Degrees.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Radians</td>
<td>10273</td>
<td>Radians.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIRVDTUnits
- DAQmxSetAIRVDTUnits
- DAQmxResetAIRVDTUnits
Analog Input >> Position >> RVDT >> Sensitivity

**Data Type:** float64

**Description:** Specifies the sensitivity of the RVDT. This value is in the units you specify with [Sensitivity Units](#). Refer to the sensor documentation to determine this value.

You can get/set/reset this property using:

- [DAQmxGetAIRVDTSSensitivity](#)
- [DAQmxSetAIRVDTSSensitivity](#)
- [DAQmxResetAIRVDTSSensitivity](#)
**Analog Input >> Position >> RVDT >> Sensitivity Units**

**Data Type:** int32

**Description:** Specifies the units of Sensitivity.

**Valid values**

| DAQmx_Val_mVoltsPerVoltPerDegree | 12507 | mVolts/Volt/Degree. |
| DAQmx_Val_mVoltsPerVoltPerRadian  | 12508 | mVolts/Volt/Radian. |

You can get/set/reset this property using:

- DAQmxGetAIRVDTSensitivityUnits
- DAQmxSetAIRVDTSensitivityUnits
- DAQmxResetAIRVDTSensitivityUnits
**Analog Input >> Sound Pressure >> Maximum Sound Pressure Level**

**Data Type:** float64

**Description:** 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 [Maximum Value](#) and [Minimum Value](#) for the channel.

You can get/set/reset this property using:

- `DAQmxGetAISoundPressureMaxSoundPressureLvl`
- `DAQmxSetAISoundPressureMaxSoundPressureLvl`
- `DAQmxResetAISoundPressureMaxSoundPressureLvl`
**Analog Input >> Sound Pressure >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return sound pressure measurements from the channel.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Pascals</td>
<td>10081</td>
<td>Pascals.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAISoundPressureUnits
- DAQmxSetAISoundPressureUnits
- DAQmxResetAISoundPressureUnits
Analog Input >> Sound Pressure >> dB Reference

**Data Type:** float64

**Description:** 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 Maximum Sound Pressure Level to a voltage level.

You can get/set/reset this property using:

- DAQmxGetAISoundPressuredBRef
- DAQmxSetAISoundPressuredBRef
- DAQmxResetAISoundPressuredBRef
**Analog Input >> Sound Pressure >> Microphone >> Sensitivity**

**Data Type:** float64

**Description:** Specifies the sensitivity of the microphone. This value is in mV/Pa. Refer to the sensor documentation to determine this value.

You can get/set/reset this property using:

- DAQmxGetAIMicrophoneSensitivity
- DAQmxSetAIMicrophoneSensitivity
- DAQmxResetAIMicrophoneSensitivity
**Analog Input >> Acceleration >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return acceleration measurements from the channel.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value (decimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AccelUnit_g</td>
<td>10186</td>
<td>1 g is approximately equal to 9.81 m/s/s.</td>
</tr>
<tr>
<td>DAQmx_Val_MetersPerSecondSquared</td>
<td>12470</td>
<td>Meters per second per second.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAIAccelUnits](#)
- [DAQmxSetAIAccelUnits](#)
- [DAQmxResetAIAccelUnits](#)
Analog Input >> Acceleration >> dB Reference

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAIAcceldBRef`
- `DAQmxSetAIAcceldBRef`
- `DAQmxResetAIAcceldBRef`
**Analog Input >> Acceleration >> Accelerometer >> Sensitivity**

**Data Type:** float64

**Description:** Specifies the sensitivity of the accelerometer. This value is in the units you specify with **Sensitivity Units**. Refer to the sensor documentation to determine this value.

You can get/set/reset this property using:

- `DAQmxGetAIAccelSensitivity`
- `DAQmxSetAIAccelSensitivity`
- `DAQmxResetAIAccelSensitivity`
Analog Input >> Acceleration >> Accelerometer >> Sensitivity Units

**Data Type:** int32  
**Description:** Specifies the units of [Sensitivity](#).

### Valid values

| DAQmx_Val_mVoltsPerG | 12509 mVolts/g. |
| DAQmx_Val_VoltsPerG  | 12510 Volts/g.  |

You can get/set/reset this property using:

- DAQmxGetAIAccelSensitivityUnits
- DAQmxSetAIAccelSensitivityUnits
- DAQmxResetAIAccelSensitivityUnits
**Analog Input >> TEDS >> Is TEDS**

**Data Type:** bool32

**Description:** Indicates if the virtual channel was initialized using a TEDS bitstream from the corresponding physical channel.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetAIIsTEDS`
Analog Input >> TEDS >> Units

Data Type: char*

Description: Indicates the units defined by TEDS information associated with the channel.

Restrictions: Not Settable

You can get this property using:

DAQmxGetAITEDSUnits
Analog Input >> General Properties >> Input Configuration >> Coupling

**Data Type:** int32

**Description:** Specifies the coupling for the channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Remove the DC offset from the signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Allow NI-DAQmx to measure all of the signal.</td>
</tr>
<tr>
<td>DAQmx_Val_GND</td>
<td>10066</td>
<td>Remove the signal from the measurement and measure only ground.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAICoupling`
- `DAQmxSetAICoupling`
- `DAQmxResetAICoupling`
**Analog Input >> General Properties >> Input Configuration >> Impedance**

**Data Type:** float64  
**Description:** Specifies the input impedance of the channel.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Impedance</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50 Ohms.</td>
</tr>
<tr>
<td>75</td>
<td>75 Ohms.</td>
</tr>
<tr>
<td>1000000</td>
<td>1 M Ohm.</td>
</tr>
<tr>
<td>10000000000</td>
<td>10 G Ohm.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIImpedance
- DAQmxSetAIImpedance
- DAQmxResetAIImpedance
Analog Input >> General Properties >> Input Configuration >> Terminal Configuration

Data Type: int32
Description: Specifies the terminal configuration for the channel.

Valid values

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_RSE</td>
<td>10083</td>
<td>Referenced Single-Ended.</td>
</tr>
<tr>
<td>DAQmx_Val_NRSE</td>
<td>10078</td>
<td>Non-Referenced Single-Ended.</td>
</tr>
<tr>
<td>DAQmx_Val_Diff</td>
<td>10106</td>
<td>Differential.</td>
</tr>
<tr>
<td>DAQmx_Val_PseudoDiff</td>
<td>12529</td>
<td>Pseudodifferential.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAITermCfg
DAQmxSetAITermCfg
DAQmxResetAITermCfg
Analog Input >> General Properties >> Input Configuration >> Input Source

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAIInputSrc
- DAQmxSetAIInputSrc
- DAQmxResetAIInputSrc
Analog Input >> General Properties >> Signal Conditioning >> Resistance Configuration

**Data Type:** int32

**Description:** 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.

### Valid values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_2Wire</td>
<td>2</td>
</tr>
<tr>
<td>DAQmx_Val_3Wire</td>
<td>3</td>
</tr>
<tr>
<td>DAQmx_Val_4Wire</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2-wire mode.</td>
</tr>
<tr>
<td></td>
<td>3-wire mode.</td>
</tr>
<tr>
<td></td>
<td>4-wire mode.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAIResistanceCfg`
- `DAQmxSetAIResistanceCfg`
- `DAQmxResetAIResistanceCfg`
Analog Input >> General Properties >> Signal Conditioning >> Lead Wire Resistance

Data Type: float64
Description: Specifies in ohms the resistance of the wires that lead to the sensor.

You can get/set/reset this property using:

DAQmxGetAILeadWireResistance
DAQmxSetAILeadWireResistance
DAQmxResetAILeadWireResistance
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Configuration

Data Type:  int32
Description:  Specifies the type of Wheatstone bridge that the sensor is.

Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FullBridge</td>
<td>Sensor is a full bridge. If you set Use Excitation 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.</td>
</tr>
<tr>
<td>DAQmx_Val_HalfBridge</td>
<td>Sensor is a half bridge. If you set Use Excitation 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.</td>
</tr>
<tr>
<td>DAQmx_Val_QuarterBridge</td>
<td>Sensor is a quarter bridge. If you set Use Excitation 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.</td>
</tr>
<tr>
<td>DAQmx_Val_NoBridge</td>
<td>Sensor is not a Wheatstone bridge.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAIBridgeCfg
DAQmxSetAIBridgeCfg
DAQmxResetAIBridgeCfg
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Nominal Resistance

**Data Type:** float64

**Description:** Specifies in ohms the resistance across each arm of the bridge in an unloaded position.

You can get/set/reset this property using:

- `DAQmxGetAIBridgeNomResistance`
- `DAQmxSetAIBridgeNomResistance`
- `DAQmxResetAIBridgeNomResistance`
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Initial Bridge Voltage

**Data Type:** float64

**Description:** Specifies in volts the output voltage of the bridge in the unloaded condition. NI-DAQmx subtracts this value from any measurements before applying scaling equations.

You can get/set/reset this property using:

- `DAQmxGetAIBridgeInitialVoltage`
- `DAQmxSetAIBridgeInitialVoltage`
- `DAQmxResetAIBridgeInitialVoltage`
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Shunt Cal Enable

Data Type: bool32
Description: Specifies whether to enable a shunt calibration switch. Use Shunt Cal Select to select the switch(es) to enable.

You can get/set/reset this property using:

DAQmxGetAIBridgeShuntCalEnable
DAQmxSetAIBridgeShuntCalEnable
DAQmxResetAIBridgeShuntCalEnable
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Shunt Cal Select

Data Type: int32

Description: Specifies which shunt calibration switch(es) to enable. Use Shunt Cal Enable to enable the switch(es) you specify with this property.

Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_A</td>
<td>12513</td>
<td>Switch A.</td>
</tr>
<tr>
<td>DAQmx_Val_B</td>
<td>12514</td>
<td>Switch B.</td>
</tr>
<tr>
<td>DAQmx_Val_AandB</td>
<td>12515</td>
<td>Switches A and B.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAIBridgeShuntCalSelect
DAQmxSetAIBridgeShuntCalSelect
DAQmxResetAIBridgeShuntCalSelect
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Gain Adjustment

**Data Type:** float64

**Description:** Specifies the result of a shunt calibration. NI-DAQmx multiplies data read from the channel by the value of this property. This value should be close to 1.0.

You can get/set/reset this property using:

- DAQmxGetAIBridgeShuntCalGainAdjust
- DAQmxSetAIBridgeShuntCalGainAdjust
- DAQmxResetAIBridgeShuntCalGainAdjust
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Coarse Potentiometer

**Data Type:** int32

**Description:** Specifies by how much to compensate for offset in the signal. This value can be between 0 and 127.

You can get/set/reset this property using:

- `DAQmxGetAIBridgeBalanceCoarsePot`
- `DAQmxSetAIBridgeBalanceCoarsePot`
- `DAQmxResetAIBridgeBalanceCoarsePot`
Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Fine Potentiometer

**Data Type:** int32

**Description:** Specifies by how much to compensate for offset in the signal. This value can be between 0 and 4095.

You can get/set/reset this property using:

- `DAQmxGetAIBridgeBalanceFinePot`
- `DAQmxSetAIBridgeBalanceFinePot`
- `DAQmxResetAIBridgeBalanceFinePot`
**Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Location**

*Data Type:* int32  
*Description:* Specifies the shunt resistor location for current measurements.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Internal</th>
<th>DAQmx_Val_External</th>
<th>10200</th>
<th>10167</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Use the built-in shunt resistor of the device.</td>
<td>Use a shunt resistor external to the device. You must specify the value of the shunt resistor by using <a href="#">Value</a>.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAICurrentShuntLoc
- DAQmxSetAICurrentShuntLoc
- DAQmxResetAICurrentShuntLoc
Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Value

**Data Type:** float64

**Description:** Specifies in ohms the external shunt resistance for current measurements.

You can get/set/reset this property using:

- DAQmxGetAICurrentShuntResistance
- DAQmxSetAICurrentShuntResistance
- DAQmxResetAICurrentShuntResistance
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Source

Data Type: int32
Description: Specifies the source of excitation.

Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value (int)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Internal</td>
<td>10200</td>
<td>Use the built-in excitation source of the device. If you select this value, you must specify the amount of excitation.</td>
</tr>
<tr>
<td>DAQmx_Val_External</td>
<td>10167</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Supply no excitation to the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIExcitSrc
- DAQmxSetAIExcitSrc
- DAQmxResetAIExcitSrc
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Value

**Data Type:** float64

**Description:** Specifies the amount of excitation that the sensor requires. If Voltage or Current is DAQmx_Val_Voltage, this value is in volts. If Voltage or Current is DAQmx_Val_Current, this value is in amperes.

You can get/set/reset this property using:

- DAQmxGetAIExcitVal
- DAQmxSetAIExcitVal
- DAQmxResetAIExcitVal
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Use Excitation For Scaling

**Data Type:** bool32

**Description:** 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 Maximum Value and Minimum Value to reflect the scaling.

For example, if you expect to acquire a voltage between -5 and 5, and you use an excitation of .10 volts to scale the measurement, set Minimum Value to -50 and set Maximum Value to 50. If you set Configuration to DAQmx_Val_NoBridge, this property has no effect on the measurement.

You can get/set/reset this property using:

- DAQmxGetAIExcitUseForScaling
- DAQmxSetAIExcitUseForScaling
- DAQmxResetAIExcitUseForScaling
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Use Multiplexed Excitation

Data Type: bool32

Description: Specifies if the SCXI-1122 multiplexes the excitation to the upper half of the channels as it advances through the scan list.

You can get/set/reset this property using:

DAQmxGetAIExcitUseMultiplexed
DAQmxSetAIExcitUseMultiplexed
DAQmxResetAIExcitUseMultiplexed
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Actual Excitation Value

Data Type: float64
Description: 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.

You can get/set/reset this property using:

DAQmxGetAIExcitActualVal
DAQmxSetAIExcitActualVal
DAQmxResetAIExcitActualVal
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> DC or AC

**Data Type:** int32
**Description:** Specifies if the excitation supply is DC or AC.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_DC</th>
<th>10050</th>
<th>DC excitation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AC</td>
<td>10045</td>
<td>AC excitation.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAIExcitDCorAC`
- `DAQmxSetAIExcitDCorAC`
- `DAQmxResetAIExcitDCorAC`
**Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Voltage or Current**

**Data Type:** int32

**Description:** Specifies if the channel uses current or voltage excitation.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Voltage</td>
<td>10322</td>
<td>Voltage excitation.</td>
</tr>
<tr>
<td>DAQmx_Val_Current</td>
<td>10134</td>
<td>Current excitation.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAIExcitVoltageOrCurrent`
- `DAQmxSetAIExcitVoltageOrCurrent`
- `DAQmxResetAIExcitVoltageOrCurrent`
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> AC Excitation >> Frequency

**Data Type:** float64

**Description:** Specifies the AC excitation frequency in Hertz.

You can get/set/reset this property using:

- `DAQmxGetAIACExcitFreq`
- `DAQmxSetAIACExcitFreq`
- `DAQmxResetAIACExcitFreq`
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> AC Excitation >> Synchronization Enable

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAIACExcitSyncEnable
- DAQmxSetAIACExcitSyncEnable
- DAQmxResetAIACExcitSyncEnable
Analog Input >> General Properties >> Signal Conditioning >> Excitation >> AC Excitation >> Wire Mode

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_4Wire</th>
<th>4-wire.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_5Wire</td>
<td>5-wire.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAIACExcitWireMode`
- `DAQmxSetAIACExcitWireMode`
- `DAQmxResetAIACExcitWireMode`
Analog Input >> General Properties >> Signal Conditioning >> Probe/Attenuator >> Attenuation Value

Data Type: float64
Description: Specifies the amount of attenuation to use.

You can get/set/reset this property using:

- DAQmxGetAIAtten
- DAQmxSetAIAtten
- DAQmxResetAIAtten
Analog Input >> General Properties >> Filter >> Analog Lowpass >> Enable

**Data Type:** bool32

**Description:** Specifies whether to enable the lowpass filter of the channel.

You can get/set/reset this property using:

- `DAQmxGetAILowpassEnable`
- `DAQmxSetAILowpassEnable`
- `DAQmxResetAILowpassEnable`
Analog Input >> General Properties >> Filter >> Analog Lowpass >> Cutoff Frequency

**Data Type:** float64

**Description:** Specifies the frequency in Hertz that corresponds to the -3dB cutoff of the filter.

You can get/set/reset this property using:

- `DAQmxGetAILowpassCutoffFreq`
- `DAQmxSetAILowpassCutoffFreq`
- `DAQmxResetAILowpassCutoffFreq`
Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> Clock Source

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>Valid value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Internal</td>
<td>10200</td>
<td>Internal to the device.</td>
</tr>
<tr>
<td>DAQmx_Val_External</td>
<td>10167</td>
<td>External to the device.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAILowpassSwitchCapClkSrc
- DAQmxSetAILowpassSwitchCapClkSrc
- DAQmxResetAILowpassSwitchCapClkSrc
Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Frequency

**Data Type:** float64

**Description:** Specifies the frequency of the external clock when you set Clock Source to DAQmx_Val_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.

You can get/set/reset this property using:

- `DAQmxGetAILowpassSwitchCapExtClkFreq`
- `DAQmxSetAILowpassSwitchCapExtClkFreq`
- `DAQmxResetAILowpassSwitchCapExtClkFreq`
Analog Input >> General Properties >> Filter >>
Analog Lowpass >> Advanced >> Switched Capacitor
>> External Clock Divisor

**Data Type:** uInt32

**Description:** Specifies the divisor for the external clock when you set Clock Source to DAQmx_Val_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.

You can get/set/reset this property using:

- DAQmxGetAILowpassSwitchCapExtClkDiv
- DAQmxSetAILowpassSwitchCapExtClkDiv
- DAQmxResetAILowpassSwitchCapExtClkDiv
Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> Output Clock Divisor

**Data Type:** uInt32

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAILowpassSwitchCapOutClkDiv
- DAQmxSetAILowpassSwitchCapOutClkDiv
- DAQmxResetAILowpassSwitchCapOutClkDiv
Analog Input >> General Properties >> Digitizer/ADC >> Resolution Units

Data Type: int32
Description: Indicates the units of Resolution Value.
Restrictions: Not Settable

Valid values

| DAQmx_Val_Bits | 10109 | Bits. |

You can get this property using:

DAQmxGetAIResolutionUnits
Analog Input >> General Properties >> Digitizer/ADC >> Resolution Value

**Data Type:**  float64

**Description:** Indicates the resolution of the analog-to-digital converter of the channel. This value is in the units you specify with Resolution Units.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetAIResolution
**Analog Input >> General Properties >> Digitizer/ADC >> Raw Sample Size**

**Data Type:** uInt32

**Description:** Indicates in bits the size of a raw sample from the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetAIRawSampSize`
Analog Input >> General Properties >> Digitizer/ADC >> Raw Sample Justification

**Data Type:** int32  
**Description:** Indicates the justification of a raw sample from the device.  
**Restrictions:** Not Settable

<table>
<thead>
<tr>
<th>Valid values</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_RightJustified</td>
<td>10279</td>
<td>Samples occupy the lower bits of the integer.</td>
</tr>
<tr>
<td>DAQmx_Val_LeftJustified</td>
<td>10209</td>
<td>Samples occupy the higher bits of the integer.</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetAIRawSampJustification
**Analog Input >> General Properties >>**

**Digitizer/ADC >> Timing Mode**

**Data Type:** int32  

**Description:** 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 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 device in the same task.

**Valid values**

<table>
<thead>
<tr>
<th>ADC Timing Mode</th>
<th>Valid Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_HighResolution</td>
<td>10195</td>
<td>Increases resolution and noise rejection while decreasing conversion rate.</td>
</tr>
<tr>
<td>DAQmx_Val_HighSpeed</td>
<td>14712</td>
<td>Increases conversion rate while decreasing resolution.</td>
</tr>
<tr>
<td>DAQmx_Val_Best50HzRejection</td>
<td>14713</td>
<td>Improves 50 Hz noise rejection while decreasing noise rejection at other frequencies.</td>
</tr>
<tr>
<td>DAQmx_Val_Best60HzRejection</td>
<td>14714</td>
<td>Improves 60 Hz noise rejection while decreasing noise rejection at other frequencies.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIADCTimingMode
- DAQmxSetAIADCTimingMode
- DAQmxResetAIADCTimingMode
Analog Input >> General Properties >> Digitizer/ADC >> Dither >> Enable

Data Type: bool32
Description: 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.

You can get/set/reset this property using:

- DAQmxGetAIDitherEnable
- DAQmxSetAIDitherEnable
- DAQmxResetAIDitherEnable
Analog Input >> General Properties >> Channel Calibration >> Has Valid Calibration Information

Data Type: bool32
Description: Indicates if the channel has calibration information.

Refer to the DAQmx Professional Tools Web site for more information and examples of channel calibration.

Restrictions: Not Settable

You can get this property using:

DAQmxGetAIChanCalHasValidCalInfo
Analog Input >> General Properties >> Channel Calibration >> Enable Calibration

**Data Type:** bool32

**Description:** Specifies whether to enable the channel calibration associated with the channel. Refer to the DAQmx Professional Tools Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- `DAQmxGetAIChanCalEnableCal`
- `DAQmxSetAIChanCalEnableCal`
- `DAQmxResetAIChanCalEnableCal`
Analog Input >> General Properties >> Channel Calibration >> Apply Calibration If Expired

**Data Type:** bool32

**Description:** Specifies whether to apply the channel calibration to the channel after the expiration date has passed.

Refer to the [DAQmx Professional Tools](https://www.ni.com/en-us/daqmx.html) Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- `DAQmxGetAIChanCalApplyCalIfExp`
- `DAQmxSetAIChanCalApplyCalIfExp`
- `DAQmxResetAIChanCalApplyCalIfExp`
Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Scale Type

Data Type: int32

Description: Specifies the method or equation form that the calibration scale uses.

Refer to the DAQmx Professional Tools Web site for more information and examples of channel calibration.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Polynomial</th>
<th>10449</th>
<th>Scale values by using an Nth order polynomial equation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Table</td>
<td>10450</td>
<td>Map an array of prescaled values to an array of corresponding scaled values, with all other values scaled proportionally.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td></td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAIChanCalScaleType
DAQmxSetAIChanCalScaleType
DAQmxResetAIChanCalScaleType
Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Pre-Scaled Values

**Data Type:** float64*

**Description:** Specifies the reference values collected when calibrating the channel.

Refer to the DAQmx Professional Tools Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- `DAQmxGetAIChanCalTablePreScaledVals`
- `DAQmxSetAIChanCalTablePreScaledVals`
- `DAQmxResetAIChanCalTablePreScaledVals`
Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Scaled Values

**Data Type:** float64*

**Description:** Specifies the acquired values collected when calibrating the channel. Refer to the [DAQmx Professional Tools](https://www.ni.com/daqmx) Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

[DAQmxGetAIChanCalTableScaledVals](https://www.ni.com/daqmx)

[DAQmxSetAIChanCalTableScaledVals](https://www.ni.com/daqmx)

[DAQmxResetAIChanCalTableScaledVals](https://www.ni.com/daqmx)
Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Forward Coefficients

**Data Type:** float64*

**Description:** Specifies the forward polynomial values used for calibrating the channel. Refer to the DAQmx Professional Tools Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- DAQmxGetAIChanCalPolyForwardCoeff
- DAQmxSetAIChanCalPolyForwardCoeff
- DAQmxResetAIChanCalPolyForwardCoeff
Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Reverse Coefficients

**Data Type:** float64*

**Description:** Specifies the reverse polynomial values used for calibrating the channel. Refer to the DAQmx Professional Tools Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- DAQmxGetAIChanCalPolyReverseCoeff
- DAQmxSetAIChanCalPolyReverseCoeff
- DAQmxResetAIChanCalPolyReverseCoeff
Analog Input >> General Properties >> Channel Calibration >> Operator Name

**Data Type:** char*

**Description:** Specifies the name of the operator who performed the channel calibration.

Refer to the [DAQmx Professional Tools](https://www.ni.com) Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- `DAQmxGetAIChanCalOperatorName`
- `DAQmxSetAIChanCalOperatorName`
- `DAQmxResetAIChanCalOperatorName`
Analog Input >> General Properties >> Channel Calibration >> Description

**Data Type:** char*

**Description:** Specifies the description entered for the calibration of the channel.

Refer to the [DAQmx Professional Tools](https://www.ni.com/en-us/daqmx-professional.html) Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- `DAQmxGetAIChanCalDesc`
- `DAQmxSetAIChanCalDesc`
- `DAQmxResetAIChanCalDesc`
**Analog Input >> General Properties >> Channel Calibration >> Verification >> Reference Values**

**Data Type:** float64*

**Description:** Specifies the reference values collected when verifying the calibration. NI-DAQmx stores these values as a record of calibration accuracy and does not use them in the scaling process. Refer to the [DAQmx Professional Tools](https://www.ni.com) Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- `DAQmxGetAIChanCalVerifRefVals`
- `DAQmxSetAIChanCalVerifRefVals`
- `DAQmxResetAIChanCalVerifRefVals`
Analog Input >> General Properties >> Channel Calibration >> Verification >> Acquired Values

**Data Type:** float64*

**Description:** Specifies the acquired values collected when verifying the calibration. NI-DAQmx stores these values as a record of calibration accuracy and does not use them in the scaling process.

Refer to the [DAQmx Professional Tools](https://www.ni.com/daqmx/) Web site for more information and examples of channel calibration.

You can get/set/reset this property using:

- DAQmxGetAIChanCalVerifAcqVals
- DAQmxSetAIChanCalVerifAcqVals
- DAQmxResetAIChanCalVerifAcqVals
Analog Input >> General Properties >> Advanced >> Range >> High

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAIRngHigh`
- `DAQmxSetAIRngHigh`
- `DAQmxResetAIRngHigh`
Analog Input >> General Properties >> Advanced >> Range >> Low

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAIRngLow`
- `DAQmxSetAIRngLow`
- `DAQmxResetAIRngLow`
Analog Input >> General Properties >> Advanced >> Gain >> Gain Value

**Data Type:** float64

**Description:** Specifies a gain factor to apply to the channel.

You can get/set/reset this property using:

- DAQmxGetAIGain
- DAQmxSetAIGain
- DAQmxResetAIGain
Analog Input >> General Properties >> Advanced >> Sample and Hold Enable

**Data Type:** bool32

**Description:** 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 Auto Zero Mode to perform an auto zero on the channel.

You can get/set/reset this property using:

- DAQmxGetAISampAndHoldEnable
- DAQmxSetAISampAndHoldEnable
- DAQmxResetAISampAndHoldEnable
Analog Input >> General Properties >> Advanced >> High Accuracy Settings >> Auto Zero Mode

**Data Type:** int32

**Description:** Specifies how often to measure ground. NI-DAQmx subtracts the measured ground voltage from every sample.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Do not perform an autozero.</td>
</tr>
<tr>
<td>DAQmx_Val.Once</td>
<td>10244</td>
<td>Perform an auto zero at the beginning of the acquisition.</td>
</tr>
<tr>
<td>DAQmx_Val.EverySample</td>
<td>10164</td>
<td>Perform an auto zero at every sample of the acquisition.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAIAutoZeroMode](#)
- [DAQmxSetAIAutoZeroMode](#)
- [DAQmxResetAIAutoZeroMode](#)
## Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism

**Data Type:** int32  
**Description:** Specifies the data transfer mode for the device.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DMA 10054</td>
<td>Direct Memory Access. Data transfers take place independently from the application.</td>
</tr>
<tr>
<td>DAQmx_Val_INTERRUPTS 10204</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_PROGRAMMEDIO 10264</td>
<td>Data transfers take place when you call an NI-DAQmx Read function or an NI-DAQmx Write function.</td>
</tr>
<tr>
<td>DAQmx_Val_USBbulk 12590</td>
<td>Data transfers take place independently from the application using a USB bulk pipe.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIDataXferMech
- DAQmxSetAIDataXferMech
- DAQmxResetAIDataXferMech
### Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition

**Data Type:** int32  
**Description:** Specifies under what condition to transfer data from the onboard memory of the device to the buffer.

#### Valid values

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_OnBrdMemMoreThanHalfFull</td>
<td>10237</td>
<td>Transfer data from the device when more than half of the onboard memory of the device fills.</td>
</tr>
<tr>
<td>DAQmx_Val_OnBrdMemNotEmpty</td>
<td>10241</td>
<td>Transfer data from the device when there is data in the onboard memory.</td>
</tr>
<tr>
<td>DAQmx_Val_OnbrdMemCustomThreshold</td>
<td>12577</td>
<td>Transfer data from the device when the number of samples specified with Data Transfer Custom Threshold are in the device FIFO.</td>
</tr>
<tr>
<td>DAQmx_Val_WhenAcqComplete</td>
<td>12546</td>
<td>Transfer data when the acquisition is complete.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAIDataXferReqCond`
- `DAQmxSetAIDataXferReqCond`
- `DAQmxResetAIDataXferReqCond`
**Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Custom Threshold**

**Data Type:** uInt32

**Description:** Specifies the number of samples that must be in the FIFO to transfer data from the device if Data Transfer Request Condition is DAQmx_Val_OnbHdrMemCustomThreshold.

You can get/set/reset this property using:

- DAQmxGetAIIDataTransferCustomThreshold
- DAQmxSetAIIDataTransferCustomThreshold
- DAQmxResetAIIDataTransferCustomThreshold
Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAIMemMapEnable`
- `DAQmxSetAIMemMapEnable`
- `DAQmxResetAIMemMapEnable`
**Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Raw Data Compression Type**

**Data Type:** int32  

**Description:** Specifies the type of compression to apply to raw samples returned from the device.  
Refer to the [DAQmx Professional Tools](https://www.ni.com/products/daqmx-professional-tools) Web site for more information and examples of raw data compression and streaming to disk.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Do not compress samples.</td>
</tr>
<tr>
<td>DAQmx_Val_LosslessPacking</td>
<td>12555</td>
<td>Remove unused bits from samples. No resolution is lost.</td>
</tr>
<tr>
<td>DAQmx_Val_LossyLSBRemoval</td>
<td>12556</td>
<td>Remove unused bits from samples. Then, if necessary, remove bits from samples until the samples are the size specified with <a href="https://www.ni.com">Compressed Sample Size</a>. This compression type limits resolution to the specified sample size.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetAI RawDataCompressionType](https://www.ni.com)
- [DAQmxSetAI RawDataCompressionType](https://www.ni.com)
- [DAQmxResetAI RawDataCompressionType](https://www.ni.com)
Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Lossy LSB Removal >> Compressed Sample Size

**Data Type:** uInt32

**Description:** Specifies the number of bits to return in a raw sample when Raw Data Compression Type is set to DAQmx_Val_LossyLSBRemoval.

Refer to the DAQmx Professional Tools Web site for more information and examples of raw data compression and streaming to disk.

You can get/set/reset this property using:

- DAQmxGetAILossyLSBRemovalCompressedSampSize
- DAQmxSetAILossyLSBRemovalCompressedSampSize
- DAQmxResetAILossyLSBRemovalCompressedSampSize
Analog Input >> General Properties >> Advanced >> Device Scaling Coefficients >> Device Scaling Coefficients

Data Type: float64*

Description: 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 array corresponds to a term of the equation. For example, if index two of the array 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.

Restrictions: Not Settable

You can get this property using:

DAQmxGetAIDevScalingCoeff
**Analog Input >> General Properties >> Advanced >> Enhanced Alias Rejection Enable**

**Data Type:** bool32

**Description:** Specifies whether to enable enhanced alias rejection. By default, enhanced alias rejection is enabled on supported devices. Leave this property set to the default value for most applications.

You can get/set/reset this property using:

- DAQmxGetAIEnhancedAliasRejectionEnable
- DAQmxSetAIEnhancedAliasRejectionEnable
- DAQmxResetAIEnhancedAliasRejectionEnable
Analog Output >> Maximum Value

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAOMax`
- `DAQmxSetAOMax`
- `DAQmxResetAOMax`
Analog Output >> Minimum Value

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

DAQmxGetAOMin

DAQmxSetAOMin

DAQmxResetAOMin
Analog Output >> Custom Scale Name

**Data Type:** char*

**Description:** Specifies the name of a custom scale for the channel.

You can get/set/reset this property using:

- DAQmxGetAOCustomScaleName
- DAQmxSetAOCustomScaleName
- DAQmxResetAOCustomScaleName
Analog Output >> Output Type

Data Type: int32
Description: Indicates whether the channel generates voltage, current, or a waveform.
Restrictions: Not Settable

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Voltage</th>
<th>10322</th>
<th>Voltage generation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Current</td>
<td>10134</td>
<td>Current generation.</td>
</tr>
<tr>
<td>DAQmx_Val_FuncGen</td>
<td>14750</td>
<td>Function generation.</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetAOOutputType
Analog Output >> Voltage >> Units

**Data Type:** int32

**Description:** Specifies in what units to generate voltage on the channel. Write data to the channel in the units you select.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Volts</th>
<th>10348</th>
<th>Volts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a [custom scale] specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAOVoltageUnits
- DAQmxSetAOVoltageUnits
- DAQmxResetAOVoltageUnits
Analog Output >> Voltage >> Current Limit

**Data Type:** float64

**Description:** Specifies the current limit, in amperes, for the voltage channel.

You can get/set/reset this property using:

- `DAQmxGetAOVoltageCurrentLimit`
- `DAQmxSetAOVoltageCurrentLimit`
- `DAQmxResetAOVoltageCurrentLimit`
**Analog Output >> Current >> Units**

**Data Type:** int32

**Description:** Specifies in what units to generate current on the channel. Write data to the channel in the units you select.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Amps</th>
<th>10342</th>
<th>Amperes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a <a href="#">custom scale</a> specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAOCurrentUnits
- DAQmxSetAOCurrentUnits
- DAQmxResetAOCurrentUnits
**Analog Output >> Function Generation >> Type**

**Data Type:** int32

**Description:** Specifies the kind of the waveform to generate.

### Valid values

| DAQmx_Val_Sine     | 14751 | Sine wave.  |
| DAQmx_Val_Triangle | 14752 | Triangle wave. |
| DAQmx_Val_Square   | 14753 | Square wave. |
| DAQmx_Val_Sawtooth | 14754 | Sawtooth wave. |

You can get/set/reset this property using:

- DAQmxGetAOFuncGenType
- DAQmxSetAOFuncGenType
- DAQmxResetAOFuncGenType
Analog Output >> Function Generation >> Frequency

**Data Type:** float64

**Description:** Specifies the frequency of the waveform to generate in hertz.

You can get/set/reset this property using:

- `DAQmxGetAOFuncGenFreq`
- `DAQmxSetAOFuncGenFreq`
- `DAQmxResetAOFuncGenFreq`
Analog Output >> Function Generation >> Amplitude

**Data Type:** float64

**Description:** Specifies the zero-to-peak amplitude of the waveform to generate in volts. Zero and negative values are valid.

You can get/set/reset this property using:

- `DAQmxGetAOFuncGenAmplitude`
- `DAQmxSetAOFuncGenAmplitude`
- `DAQmxResetAOFuncGenAmplitude`
Analog Output >> Function Generation >> Offset

**Data Type:** float64  

**Description:** Specifies the voltage offset of the waveform to generate.

You can get/set/reset this property using:

- DAQmxGetAOFuncGenOffset
- DAQmxSetAOFuncGenOffset
- DAQmxResetAOFuncGenOffset
Analog Output >> Function Generation >> Square >> DutyCycle

**Data Type:** float64

**Description:** Specifies the square wave duty cycle of the waveform to generate.

You can get/set/reset this property using:

- DAQmxGetAOFuncGenSquareDutyCycle
- DAQmxSetAOFuncGenSquareDutyCycle
- DAQmxResetAOFuncGenSquareDutyCycle
Analog Output >> Function Generation >> Modulation >> Type

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AM</td>
<td>14756</td>
<td>Amplitude modulation.</td>
</tr>
<tr>
<td>DAQmx_Val_FM</td>
<td>14757</td>
<td>Frequency modulation.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>No modulation.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAOFuncGenModulationType`
- `DAQmxSetAOFuncGenModulationType`
- `DAQmxResetAOFuncGenModulationType`
Analog Output >> Function Generation >> Modulation >> FM Deviation

**Data Type:** float64

**Description:** Specifies the FM deviation in hertz per volt when `Type` is DAQmx_Val_FM.

You can get/set/reset this property using:

- `DAQmxGetAOFuncGenFMDeviation`
- `DAQmxSetAOFuncGenFMDeviation`
- `DAQmxResetAOFuncGenFMDeviation`
**Analog Output >> General Properties >> Output Configuration >> Output Impedance**

**Data Type:** float64  
**Description:** Specifies in ohms the impedance of the analog output stage of the device.

You can get/set/reset this property using:

- `DAQmxGetAOOutputImpedance`
- `DAQmxSetAOOutputImpedance`
- `DAQmxResetAOOutputImpedance`
Analog Output >> General Properties >> Output Configuration >> Load Impedance

Data Type: float64

Description: Specifies in ohms the load impedance connected to the analog output channel.

You can get/set/reset this property using:

- DAQmxGetAOLoadImpedance
- DAQmxSetAOLoadImpedance
- DAQmxResetAOLoadImpedance
Analog Output >> General Properties >> Output Configuration >> Idle Output Behavior

**Data Type:** int32

**Description:** Specifies the state of the channel when no generation is in progress.

### Valid values

<table>
<thead>
<tr>
<th>Data Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ZeroVolts</td>
<td>12526</td>
<td>Generate 0 V.</td>
</tr>
<tr>
<td>DAQmx_Val_HighImpedance</td>
<td>12527</td>
<td>Set the channel to high impedance, effectively disconnecting the analog output circuitry from the I/O connector.</td>
</tr>
<tr>
<td>DAQmx_Val_MaintainExistingValue</td>
<td>12528</td>
<td>Continue generating the current value.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAOIdleOutputBehavior
- DAQmxSetAOIdleOutputBehavior
- DAQmxResetAOIdleOutputBehavior
Analog Output >> General Properties >> Output Configuration >> Terminal Configuration

Data Type: int32
Description: Specifies the terminal configuration of the channel.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_RSE</th>
<th>10083</th>
<th>Referenced Single-Ended.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Diff</td>
<td>10106</td>
<td>Differential.</td>
</tr>
<tr>
<td>DAQmx_Val_PseudoDiff</td>
<td>12529</td>
<td>Pseudodifferential.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAOTermCfg`
- `DAQmxSetAOTermCfg`
- `DAQmxResetAOTermCfg`
Analog Output >> General Properties >> DAC >> Resolution Units

Data Type: int32

Description: Specifies the units of Resolution Value.

Valid values

| DAQmx_Val_Bits   | 10109 | Bits. |

You can get/set/reset this property using:

- DAQmxGetAOResolutionUnits
- DAQmxSetAOResolutionUnits
- DAQmxResetAOResolutionUnits
Analog Output >> General Properties >> DAC >> Resolution Value

**Data Type:**  float64

**Description:** Indicates the resolution of the digital-to-analog converter of the channel. This value is in the units you specify with Resolution Units.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetAOResolution](https://example.com/DaqmxGetAOResolution)
Analog Output >> General Properties >> DAC >> Range >> High

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAODACRngHigh
- DAQmxSetAODACRngHigh
- DAQmxResetAODACRngHigh
Analog Output >> General Properties >> DAC >> Range >> Low

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAODACRngLow`
- `DAQmxSetAODACRngLow`
- `DAQmxResetAODACRngLow`
Analog Output >> General Properties >> DAC >> Reference Voltage >> Connect DAC Reference to Ground

**Data Type:** bool32

**Description:** 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 Source is DAQmx_Val_Internal and Allow Connecting DAC Reference to Ground at Runtime is TRUE.

You can get/set/reset this property using:

- `DAQmxGetAODACRefConnToGnd`
- `DAQmxSetAODACRefConnToGnd`
- `DAQmxResetAODACRefConnToGnd`
Analog Output >> General Properties >> DAC >> Reference Voltage >> Allow Connecting DAC Reference to Ground at Runtime

**Data Type:** bool32

**Description:** Specifies whether to allow grounding the internal DAC reference at run time. You must set this property to TRUE and set `Source` to `DAQmx_Val_Internal` before you can set `Connect DAC Reference to Ground` to TRUE.

You can get/set/reset this property using:

- `DAQmxGetAODACRefAllowConnToGnd`
- `DAQmxSetAODACRefAllowConnToGnd`
- `DAQmxResetAODACRefAllowConnToGnd`
Analog Output >> General Properties >> DAC >> Reference Voltage >> Source

**Data Type:** int32

**Description:** Specifies the source of the DAC reference voltage. The value of this voltage source determines the full-scale value of the DAC.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Internal</th>
<th>10200</th>
<th>Internal to the device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_External</td>
<td>10167</td>
<td>External to the device.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAODACRefSrc
- DAQmxSetAODACRefSrc
- DAQmxResetAODACRefSrc
Analog Output >> General Properties >> DAC >> Reference Voltage >> External Source

Data Type: char*

Description: Specifies the source of the DAC reference voltage if Source is DAQmx_Val_External. The valid sources for this signal vary by device.

You can get/set/reset this property using:

DAQmxGetAO DACRefExtSrc
DAQmxSetAO DACRefExtSrc
DAQmxResetAO DACRefExtSrc
**Analog Output >> General Properties >> DAC >> Reference Voltage >> Value**

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAORefVal
- DAQmxSetAORefVal
- DAQmxResetAORefVal
Analog Output >> General Properties >> DAC >> Offset Voltage >> Source

Data Type: int32

Description: Specifies the source of the DAC offset voltage. The value of this voltage source determines the full-scale value of the DAC.

Valid values

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal to the device.</td>
<td>DAQmx_Val_Internal 10200</td>
</tr>
<tr>
<td>External to the device.</td>
<td>DAQmx_Val_External 10167</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAODACOffsetSrc
- DAQmxSetAODACOffsetSrc
- DAQmxResetAODACOffsetSrc
Analog Output >> General Properties >> DAC >> Offset Voltage >> External Source

**Data Type:** char*

**Description:** Specifies the source of the DAC offset voltage if `source` is DAQmx_Val_External. The valid sources for this signal vary by device.

You can get/set/reset this property using:

- `DAQmxGetAODACOffsetExtSrc`
- `DAQmxSetAODACOffsetExtSrc`
- `DAQmxResetAODACOffsetExtSrc`
Analog Output >> General Properties >> DAC >> Offset Voltage >> Value

Data Type:  float64

Description:  Specifies in volts the value of the DAC offset voltage. To achieve best accuracy, the DAC offset value should be hand calibrated.

You can get/set/reset this property using:

DAQmxGetAOOffsetValue
DAQmxSetAOOffsetValue
DAQmxResetAOOffsetValue
Analog Output >> General Properties >> DAC >> Reglitching Enable

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAOReglitchEnable`
- `DAQmxSetAOReglitchEnable`
- `DAQmxResetAOReglitchEnable`
Analog Output >> General Properties >> Advanced >> Gain >> Gain Value

**Data Type:** float64

**Description:** Specifies in decibels the gain factor to apply to the channel.

You can get/set/reset this property using:

- DAQmxGetAOGain
- DAQmxSetAOGain
- DAQmxResetAOGain
Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAOUseOnlyOnBrdMem`
- `DAQmxSetAOUseOnlyOnBrdMem`
- `DAQmxResetAOUseOnlyOnBrdMem`
Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism

Data Type: int32
Description: Specifies the data transfer mode for the device.

Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DMA</td>
<td>Direct Memory Access. Data transfers take place independently from the application.</td>
</tr>
<tr>
<td>DAQmx_Val_Interrupts</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_ProgrammedIO</td>
<td>Data transfers take place when you call an NI-DAQmx Read function or an NI-DAQmx Write function.</td>
</tr>
<tr>
<td>DAQmx_Val_USBbulk</td>
<td>Data transfers take place independently from the application using a USB bulk pipe.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAODataXferMech
DAQmxSetAODataXferMech
DAQmxResetAODataXferMech
Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition

Data Type: int32
Description: Specifies under what condition to transfer data from the buffer to the onboard memory of the device.

Valid values

| DAQmx_Val_OnBrdMemEmpty       | 10235 | Transfer data to the device only when there is no data in the onboard memory of the device. |
| DAQmx_Val_OnBrdMemHalfFullOrLess | 10239 | Transfer data to the device any time the onboard memory is less than half full. |
| DAQmx_Val_OnBrdMemNotFull     | 10242 | Transfer data to the device any time the onboard memory of the device is not full. |

You can get/set/reset this property using:

DAQmxGetAODataXferReqCond
DAQmxSetAODataXferReqCond
DAQmxResetAODataXferReqCond
Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetAOMemMapEnable`
- `DAQmxSetAOMemMapEnable`
- `DAQmxResetAOMemMapEnable`
Data Type: float64*

Description: 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 array corresponds to a term of the equation. For example, if index two of the array is 4, the third term of the equation is $4x^2$. Scaling coefficients do not account for any custom scales that may be applied to the channel.

Restrictions: Not Settable

You can get this property using:

DAQmxGetAODevScalingCoeff
**Analog Output >> General Properties >> Advanced >> Enhanced Image Rejection Enable**

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetAOEnhancedImageRejectionEnable
- DAQmxSetAOEnhancedImageRejectionEnable
- DAQmxResetAOEnhancedImageRejectionEnable
Digital Input >> Invert Lines

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetDIInvertLines
- DAQmxSetDIInvertLines
- DAQmxResetDIInvertLines
Digital Input >> Number of Lines

Data Type: uInt32
Description: Indicates the number of digital lines in the channel.
Restrictions: Not Settable

You can get this property using:

DAQmxGetDINumLines
Digital Input >> Digital Filter >> Enable

Data Type: bool32

Description: 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.

You can get/set/reset this property using:

- DAQmxGetDIDigFltrEnable
- DAQmxSetDIDigFltrEnable
- DAQmxResetDIDigFltrEnable
Digital Input >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes as a valid high or low state transition.

You can get/set/reset this property using:

- DAQmxGetDIDigFltrMinPulseWidth
- DAQmxSetDIDigFltrMinPulseWidth
- DAQmxResetDIDigFltrMinPulseWidth
Digital Input >> Tristate

Data Type:    bool32

Description: 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.

You can get/set/reset this property using:

DAQmxGetDITristate
DAQmxSetDITristate
DAQmxResetDITristate
**Digital Input >> Logic Family**

**Data Type:** int32

**Description:** 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 device documentation for information on the logic high and logic low voltages for these logic families.

**Valid values**

<table>
<thead>
<tr>
<th>Data Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_2point5V</td>
<td>14620</td>
<td>Compatible with 2.5 V CMOS signals.</td>
</tr>
<tr>
<td>DAQmx_Val_3point3V</td>
<td>14621</td>
<td>Compatible with LV TTL signals.</td>
</tr>
<tr>
<td>DAQmx_Val_5V</td>
<td>14619</td>
<td>Compatible with TTL and 5 V CMOS signals.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDILogicFamily
- DAQmxSetDILogicFamily
- DAQmxResetDILogicFamily
### Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism

**Data Type:** int32  
**Description:** Specifies the data transfer mode for the device.

#### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_DMA</th>
<th>10054</th>
<th>Direct Memory Access. Data transfers take place independently from the application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Interrupts</td>
<td>10204</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_ProgrammedIO</td>
<td>10264</td>
<td>Data transfers take place when you call an NI-DAQmx Read function or an NI-DAQmx Write function.</td>
</tr>
<tr>
<td>DAQmx_Val_USBbulk</td>
<td>12590</td>
<td>Data transfers take place independently from the application using a USB bulk pipe.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetDIDataXferMech`
- `DAQmxSetDIDataXferMech`
- `DAQmxResetDIDataXferMech`
**Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition**

**Data Type:** int32  
**Description:** Specifies under what condition to transfer data from the onboard memory of the device to the buffer.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_OnBrdMemMoreThanHalfFull</td>
<td>10237</td>
<td>Transfer data from the device when more than half of the onboard memory of the device fills.</td>
</tr>
<tr>
<td>DAQmx_Val_OnBrdMemNotEmpty</td>
<td>10241</td>
<td>Transfer data from the device when there is data in the onboard memory.</td>
</tr>
<tr>
<td>DAQmx_Val_OnbrdMemCustomThreshold</td>
<td>12577</td>
<td>Transfer data from the device when the number of samples specified with <a href="#">Data Transfer Custom Threshold</a> are in the device FIFO.</td>
</tr>
<tr>
<td>DAQmx_Val_WhenAcqComplete</td>
<td>12546</td>
<td>Transfer data when the acquisition is complete.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- [DAQmxGetDIDataXferReqCond](#)  
- [DAQmxSetDIDataXferReqCond](#)  
- [DAQmxResetDIDataXferReqCond](#)
Digital Input >> General Properties >> Advanced >>
Data Transfer and Memory >> Memory Mapping for
Programmed IO Enable

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetDIMemMapEnable`
- `DAQmxSetDIMemMapEnable`
- `DAQmxResetDIMemMapEnable`
Digital Input >> General Properties >> Advanced >> Acquire On

**Data Type:** int32  
**Description:** Specifies on which edge of the sample clock to acquire samples.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_SampClkActiveEdge</td>
<td>14617</td>
</tr>
<tr>
<td>DAQmx_Val_SampClkInactiveEdge</td>
<td>14618</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDIAcquireOn
- DAQmxSetDIAcquireOn
- DAQmxResetDIAcquireOn
**Digital Output >> Output Drive Type**

**Data Type:** int32  
**Description:** Specifies the drive type for digital output channels.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveDrive 12573</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_OpenCollector 12574</td>
<td>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.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDOOutputDriveType
- DAQmxSetDOOutputDriveType
- DAQmxResetDOOutputDriveType
Digital Output >> Invert Lines

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetDOInvertLines`
- `DAQmxSetDOInvertLines`
- `DAQmxResetDOInvertLines`
Digital Output >> Number of Lines

**Data Type:** uInt32

**Description:** Indicates the number of digital lines in the channel.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDONumLines](#)
Digital Output >> Tristate

**Data Type:** bool32

**Description:** 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.

- Setting this property before you commit the task determines whether NI-DAQmx drives data on the channel or tristates the channel after the task transitions to the committed state.
- Use this property when you want to read and write to channels in a task. Set this property to TRUE before you read from the channel and set this property to FALSE before you write to the channel.
- When you read from an output channel that is not tristated, the value corresponds to the current value you are driving on the channel.
- When you write to a channel that is tristated, no change occurs until you set this property to FALSE. After you set this property to FALSE, the value NI-DAQmx drives on the channel corresponds to the last value written to the channel.
- Ensure the channel is tristated before any external devices that you connect to the channel drive data onto the channel. Failure to do so could result in double-driving the channel and damaging the device.
- This property is supported only for line-configurable, bidirectional ports.

You can get/set/reset this property using:

- `DAQmxGetDOTristate`
- `DAQmxSetDOTristate`
- `DAQmxResetDOTristate`
Digital Output >> Line States >> Start State

**Data Type:** int32

**Description:** Specifies the state of the lines in a digital output task when the task starts.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>Logic high.</td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Logic low.</td>
</tr>
<tr>
<td>DAQmx_Val_Tristate</td>
<td>10310</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_NoChange</td>
<td>10160</td>
<td>Do not change the state of the lines. On some devices, you can select this value only for entire ports.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDOLineStatesStartState
- DAQmxSetDOLineStatesStartState
- DAQmxResetDOLineStatesStartState
Digital Output >> Line States >> Paused State

Data Type: int32

Description: Specifies the state of the lines in a digital output task when the task pauses.

Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>Logic high.</td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Logic low.</td>
</tr>
<tr>
<td>DAQmx_Val_Tristate</td>
<td>10310</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_NoChange</td>
<td>10160</td>
<td>Do not change the state of the lines. On some devices, you can select this value only for entire ports.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDOLineStatesPausedState
- DAQmxSetDOLineStatesPausedState
- DAQmxResetDOLineStatesPausedState
Digital Output >> Line States >> Done State

**Data Type:** int32

**Description:** Specifies the state of the lines in a digital output task when the task completes execution.

<table>
<thead>
<tr>
<th>Valid values</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>Logic high.</td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Logic low.</td>
</tr>
<tr>
<td>DAQmx_Val_Tristate</td>
<td>10310</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_NoChange</td>
<td>10160</td>
<td>Do not change the state of the lines. On some devices, you can select this value only for entire ports.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDOLineStatesDoneState
- DAQmxSetDOLineStatesDoneState
- DAQmxResetDOLineStatesDoneState
Digital Output >> Logic Family

**Data Type:** int32

**Description:** 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 device documentation for information on the logic high and logic low voltages for these logic families.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_2point5V</td>
<td>14620</td>
<td>Compatible with 2.5 V CMOS signals.</td>
</tr>
<tr>
<td>DAQmx_Val_3point3V</td>
<td>14621</td>
<td>Compatible with LVTTL signals.</td>
</tr>
<tr>
<td>DAQmx_Val_5V</td>
<td>14619</td>
<td>Compatible with TTL and 5 V CMOS signals.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDOLogicFamily
- DAQmxSetDOLogicFamily
- DAQmxResetDOLogicFamily
Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetDOUseOnlyOnBrdMem`
- `DAQmxSetDOUseOnlyOnBrdMem`
- `DAQmxResetDOUseOnlyOnBrdMem`
Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism

**Data Type:** int32

**Description:** Specifies the data transfer mode for the device.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DMA</td>
<td>Direct Memory Access. Data transfers take place independently from the application.</td>
</tr>
<tr>
<td>DAQmx_Val_Interruents</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_ProgrammedIO</td>
<td>Data transfers take place when you call an NI-DAQmx Read function or an NI-DAQmx Write function.</td>
</tr>
<tr>
<td>DAQmx_Val_USBbulk</td>
<td>Data transfers take place independently from the application using a USB bulk pipe.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDODataXferMech
- DAQmxSetDODataXferMech
- DAQmxResetDODataXferMech
Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition

**Data Type:** int32

**Description:** Specifies under what condition to transfer data from the buffer to the onboard memory of the device.

**Valid values**

<table>
<thead>
<tr>
<th>Data Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_OnBrdMemEmpty</td>
<td>10235</td>
<td>Transfer data to the device only when there is no data in the onboard memory of the device.</td>
</tr>
<tr>
<td>DAQmx_Val_OnBrdMemHalfFullOrLess</td>
<td>10239</td>
<td>Transfer data to the device any time the onboard memory is less than half full.</td>
</tr>
<tr>
<td>DAQmx_Val_OnBrdMemNotFull</td>
<td>10242</td>
<td>Transfer data to the device any time the onboard memory of the device is not full.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDODataXferReqCond
- DAQmxSetDODataXferReqCond
- DAQmxResetDODataXferReqCond
Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetDOMemMapEnable`
- `DAQmxSetDOMemMapEnable`
- `DAQmxResetDOMemMapEnable`
Digital Output >> General Properties >> Advanced >> Generate On

**Data Type:** int32

**Description:** Specifies on which edge of the sample clock to generate samples.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_SampClkActiveEdge</th>
<th>14617</th>
<th>Active edges.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_SampClkInactiveEdge</td>
<td>14618</td>
<td>Inactive edges.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetDOGenerateOn`
- `DAQmxSetDOGenerateOn`
- `DAQmxResetDOGenerateOn`
**Counter Input >> Maximum Value**

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetCIMax
- DAQmxSetCIMax
- DAQmxResetCIMax
Counter Input >> Minimum Value

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetCIMin
- DAQmxSetCIMin
- DAQmxResetCIMin
Counter Input >> Custom Scale Name

Data Type: char*

Description: Specifies the name of a custom scale for the channel.

You can get/set/reset this property using:

DAQmxGetCICustomScaleName
DAQmxSetCICustomScaleName
DAQmxResetCICustomScaleName
### Counter Input >> Measurement Type

**Data Type:** int32  
**Description:** Indicates the measurement to take with the channel.  
**Restrictions:** Not Settable

#### Valid values

<table>
<thead>
<tr>
<th>Measurement Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_CountEdges</td>
<td>10125</td>
<td>Count edges of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_Freq</td>
<td>10179</td>
<td>Measure the frequency of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_Period</td>
<td>10256</td>
<td>Measure the period of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_PulseWidth</td>
<td>10359</td>
<td>Measure the width of a pulse of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_SemiPeriod</td>
<td>10289</td>
<td>Measure the time between state transitions of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_Position_AngEncoder</td>
<td>10360</td>
<td>Angular position measurement using an angular encoder.</td>
</tr>
<tr>
<td>DAQmx_Val_Position_LinEncoder</td>
<td>10361</td>
<td>Linear position measurement using a linear encoder.</td>
</tr>
<tr>
<td>DAQmx_Val_TwoEdgeSep</td>
<td>10267</td>
<td>Measure time between edges of two digital signals.</td>
</tr>
<tr>
<td>DAQmx_Val_GPS_Timestamp</td>
<td>10362</td>
<td>Timestamp measurement, synchronizing the counter to a GPS receiver.</td>
</tr>
</tbody>
</table>

You can get this property using:  
DAQmxGetCIMeasType
**Counter Input >> Frequency >> Units**

**Data Type:** int32  
**Description:** Specifies the units to use to return frequency measurements.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Hz</td>
<td>10373</td>
<td>Hertz.</td>
</tr>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIFreqUnits
- DAQmxSetCIFreqUnits
- DAQmxResetCIFreqUnits
Counter Input >> Frequency >> Input Terminal

**Data Type:**  char*

**Description:** Specifies the input terminal of the signal to measure.

You can get/set/reset this property using:

- `DAQmxGetCIFreqTerm`
- `DAQmxSetCIFreqTerm`
- `DAQmxResetCIFreqTerm`
**Counter Input >> Frequency >> Starting Edge**

**Data Type:** int32

**Description:** Specifies between which edges to measure the frequency of the signal.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIFreqStartingEdge
- DAQmxSetCIFreqStartingEdge
- DAQmxResetCIFreqStartingEdge
## Counter Input >> Frequency >> Measurement Specifications >> Method

**Data Type:** int32  
**Description:** Specifies the method to use to measure the frequency of the signal.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_LowFreq1Ctr</td>
<td>10105</td>
<td>Use one counter that uses a constant timebase to measure the input signal.</td>
</tr>
<tr>
<td>DAQmx_Val_HighFreq2Ctr</td>
<td>10157</td>
<td>Use two counters, one of which counts pulses of the signal to measure during the specified measurement time.</td>
</tr>
<tr>
<td>DAQmx_Val_LargeRng2Ctr</td>
<td>10205</td>
<td>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.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCIFreqMeasMeth`
- `DAQmxSetCIFreqMeasMeth`
- `DAQmxResetCIFreqMeasMeth`
Counter Input >> Frequency >> Measurement Specifications >> High Frequency >> Measurement Time

**Data Type:** float64

**Description:** Specifies in seconds the length of time to measure the frequency of the signal if `Method` is `DAQmx_Val_HighFreq2Ctr`. 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.

You can get/set/reset this property using:

- `DAQmxGetCIFreqMeasTime`
- `DAQmxSetCIFreqMeasTime`
- `DAQmxResetCIFreqMeasTime`
Counter Input >> Frequency >> Measurement Specifications >> Large Range >> Divisor

**Data Type:** uInt32

**Description:** Specifies the value by which to divide the input signal if Method is DAQmx_Val_LargeRng2Ctr. 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.

You can get/set/reset this property using:

- DAQmxGetCIFreqDiv
- DAQmxSetCIFreqDiv
- DAQmxResetCIFreqDiv
Counter Input >> Frequency >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetCIFreqDigFtrEnable`
- `DAQmxSetCIFreqDigFtrEnable`
- `DAQmxResetCIFreqDigFtrEnable`
**Counter Input >> Frequency >> Digital Filter >> Minimum Pulse Width**

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCIFreqDigFltrMinPulseWidth`
- `DAQmxSetCIFreqDigFltrMinPulseWidth`
- `DAQmxResetCIFreqDigFltrMinPulseWidth`
Counter Input >> Frequency >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- DAQmxGetCIFreqDigFtrTimebaseSrc
- DAQmxSetCIFreqDigFtrTimebaseSrc
- DAQmxResetCIFreqDigFtrTimebaseSrc
Counter Input >> Frequency >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCIFreqDigFltrTimebaseRate`
- `DAQmxSetCIFreqDigFltrTimebaseRate`
- `DAQmxResetCIFreqDigFltrTimebaseRate`
Counter Input >> Frequency >> Digital Synchronization >> Enable

Data Type: bool32

Description: Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

DAQmxGetCIFreqDigSyncEnable
DAQmxSetCIFreqDigSyncEnable
DAQmxResetCIFreqDigSyncEnable
Counter Input >> Period >> Units

**Data Type:** int32

**Description:** Specifies the unit to use to return period measurements.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Seconds</td>
<td>10364</td>
<td>Seconds.</td>
</tr>
<tr>
<td>DAQmx_Val.Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIPeriodUnits
- DAQmxSetCIPeriodUnits
- DAQmxResetCIPeriodUnits
Counter Input >> Period >> Input Terminal

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to measure.

You can get/set/reset this property using:

- DAQmxGetCIPeriodTerm
- DAQmxSetCIPeriodTerm
- DAQmxResetCIPeriodTerm
Counter Input >> Period >> Starting Edge

**Data Type:** int32

**Description:** Specifies between which edges to measure the period of the signal.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIPeriodStartingEdge
- DAQmxSetCIPeriodStartingEdge
- DAQmxResetCIPeriodStartingEdge
Counter Input >> Period >> Measurement Specifications >> Method

Data Type: int32
Description: Specifies the method to use to measure the period of the signal.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_LowFreq1Ctr</th>
<th>10105</th>
<th>Use one counter that uses a constant timebase to measure the input signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_HighFreq2Ctr</td>
<td>10157</td>
<td>Use two counters, one of which counts pulses of the signal to measure during the specified measurement time.</td>
</tr>
<tr>
<td>DAQmx_Val_LargeRng2Ctr</td>
<td>10205</td>
<td>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.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIPeriodMeasMeth
- DAQmxSetCIPeriodMeasMeth
- DAQmxResetCIPeriodMeasMeth
Counter Input >> Period >> Measurement Specifications >> High Frequency >> Measurement Time

**Data Type:** float64

**Description:** Specifies in seconds the length of time to measure the period of the signal if `Method` is `DAQmx_Val_HighFreq2Ctr`. 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.

You can get/set/reset this property using:

- `DAQmxGetCIPeriodMeasTime`
- `DAQmxSetCIPeriodMeasTime`
- `DAQmxResetCIPeriodMeasTime`
Counter Input >> Period >> Measurement Specifications >> Large Range >> Divisor

**Data Type:** uInt32

**Description:** Specifies the value by which to divide the input signal if Method is DAQmx_Val_LargeRng2Ctr. 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.

You can get/set/reset this property using:

- DAQmxGetCIPeriodDiv
- DAQmxSetCIPeriodDiv
- DAQmxResetCIPeriodDiv
Data Type: bool32
Description: Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

DAQmxGetCIPeriodDigFltrEnable
DAQmxSetCIPeriodDigFltrEnable
DAQmxResetCIPeriodDigFltrEnable
Counter Input >> Period >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCIPeriodDigFltrMinPulseWidth`
- `DAQmxSetCIPeriodDigFltrMinPulseWidth`
- `DAQmxResetCIPeriodDigFltrMinPulseWidth`
Counter Input >> Period >> Digital Filter >> Timebase >> Source

**Data Type:** char

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCIPeriodDigFltrTimebaseSrc`
- `DAQmxSetCIPeriodDigFltrTimebaseSrc`
- `DAQmxResetCIPeriodDigFltrTimebaseSrc`
Counter Input >> Period >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCIPeriodDigFltrTimebaseRate`
- `DAQmxSetCIPeriodDigFltrTimebaseRate`
- `DAQmxResetCIPeriodDigFltrTimebaseRate`
Counter Input >> Period >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetCIPeriodDigSyncEnable`
- `DAQmxSetCIPeriodDigSyncEnable`
- `DAQmxResetCIPeriodDigSyncEnable`
Counter Input >> Count Edges >> Input Terminal

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to measure.

You can get/set/reset this property using:

- [DAQmxGetCICountEdgesTerm](#)
- [DAQmxSetCICountEdgesTerm](#)
- [DAQmxResetCICountEdgesTerm](#)
Counter Input >> Count Edges >> Count Direction >> Direction

**Data Type:** int32

**Description:** Specifies whether to increment or decrement the counter on each edge.

### Valid values

<table>
<thead>
<tr>
<th>Data Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_CountUp</td>
<td>10128</td>
<td>Increment counter.</td>
</tr>
<tr>
<td>DAQmx_Val_CountDown</td>
<td>10124</td>
<td>Decrement counter.</td>
</tr>
<tr>
<td>DAQmx_Val_ExtControlled</td>
<td>10326</td>
<td>The state of a digital line controls the count direction. Each counter has a default count direction terminal.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesDir`
- `DAQmxSetCICountEdgesDir`
- `DAQmxResetCICountEdgesDir`
Counter Input >> Count Edges >> Count Direction >> Terminal

**Data Type:** char*

**Description:** Specifies the source terminal of the digital signal that controls the count direction if Direction is DAQmx_Val_ExtControlled.

You can get/set/reset this property using:

- DAQmxGetCICountEdgesDirTerm
- DAQmxSetCICountEdgesDirTerm
- DAQmxResetCICountEdgesDirTerm
**Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Enable**

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- DAQmxGetCICountEdgesCountDirDigFltrEnable
- DAQmxSetCICountEdgesCountDirDigFltrEnable
- DAQmxResetCICountEdgesCountDirDigFltrEnable
Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesCountDirDigFltrMinPulseWidth`
- `DAQmxSetCICountEdgesCountDirDigFltrMinPulseWidth`
- `DAQmxResetCICountEdgesCountDirDigFltrMinPulseWidth`
Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesCountDirDigFltrTimebaseSrc`
- `DAQmxSetCICountEdgesCountDirDigFltrTimebaseSrc`
- `DAQmxResetCICountEdgesCountDirDigFltrTimebaseSrc`
**Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Rate**

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesCountDirDigFltrTimebaseRate`
- `DAQmxSetCICountEdgesCountDirDigFltrTimebaseRate`
- `DAQmxResetCICountEdgesCountDirDigFltrTimebaseRate`
**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetCICountEdgesCountDirDigSyncEnable
- DAQmxSetCICountEdgesCountDirDigSyncEnable
- DAQmxResetCICountEdgesCountDirDigSyncEnable
Counter Input >> Count Edges >> Initial Count

**Data Type:** uInt32

**Description:** Specifies the starting value from which to count.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesInitialCnt`
- `DAQmxSetCICountEdgesInitialCnt`
- `DAQmxResetCICountEdgesInitialCnt`
Counter Input >> Count Edges >> Active Edge

**Data Type:** int32

**Description:** Specifies on which edges to increment or decrement the counter.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCICountEdgesActiveEdge
- DAQmxSetCICountEdgesActiveEdge
- DAQmxResetCICountEdgesActiveEdge
Counter Input >> Count Edges >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesDigFltrEnable`
- `DAQmxSetCICountEdgesDigFltrEnable`
- `DAQmxResetCICountEdgesDigFltrEnable`
Counter Input >> Count Edges >> Digital Filter >> Minimum Pulse Width

**Data Type:**  float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesDigFltrMinPulseWidth`
- `DAQmxSetCICountEdgesDigFltrMinPulseWidth`
- `DAQmxResetCICountEdgesDigFltrMinPulseWidth`
Counter Input >> Count Edges >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesDigFltrTimebaseSrc`
- `DAQmxSetCICountEdgesDigFltrTimebaseSrc`
- `DAQmxResetCICountEdgesDigFltrTimebaseSrc`
Counter Input >> Count Edges >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- DAQmxGetCICountEdgesDigFltrTimebaseRate
- DAQmxSetCICountEdgesDigFltrTimebaseRate
- DAQmxResetCICountEdgesDigFltrTimebaseRate
**Counter Input >> Count Edges >> Digital Synchronization >> Enable**

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetCICountEdgesDigSyncEnable`
- `DAQmxSetCICountEdgesDigSyncEnable`
- `DAQmxResetCICountEdgesDigSyncEnable`
**Counter Input >> Position >> Angular Encoder >> Units**

**Data Type:** int32

**Description:** Specifies the units to use to return angular position measurements from the channel.

**Valid values**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Degrees</td>
<td>10146</td>
<td>Degrees.</td>
</tr>
<tr>
<td>DAQmx_Val_Radians</td>
<td>10273</td>
<td>Radians.</td>
</tr>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a [custom scale] specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCIAngEncoderUnits`
- `DAQmxSetCIAngEncoderUnits`
- `DAQmxResetCIAngEncoderUnits`
Counter Input >> Position >> Angular Encoder >> Pules Per Revolution

**Data Type:** uInt32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetCIAngEncoderPulsesPerRev`
- `DAQmxSetCIAngEncoderPulsesPerRev`
- `DAQmxResetCIAngEncoderPulsesPerRev`
Counter Input >> Position >> Angular Encoder >> Initial Angle

**Data Type:**  float64

**Description:** Specifies the starting angle of the encoder. This value is in the units you specify with **Units**.

You can get/set/reset this property using:

- DAQmxGetCIAngEncoderInitialAngle
- DAQmxSetCIAngEncoderInitialAngle
- DAQmxResetCIAngEncoderInitialAngle
Counter Input >> Position >> Linear Encoder >> Units

Data Type: int32

Description: Specifies the units to use to return linear encoder measurements from the channel.

Valid values

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Meters</td>
<td>10219</td>
<td>Meters.</td>
</tr>
<tr>
<td>DAQmx_Val_Inches</td>
<td>10379</td>
<td>Inches.</td>
</tr>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCILinEncoderUnits
- DAQmxSetCILinEncoderUnits
- DAQmxResetCILinEncoderUnits
Counter Input >> Position >> Linear Encoder >> Distance Per Pulse

**Data Type:** float64

**Description:** 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 Units.

You can get/set/reset this property using:

- DAQmxGetCILinEncoderDistPerPulse
- DAQmxSetCILinEncoderDistPerPulse
- DAQmxResetCILinEncoderDistPerPulse
Counter Input >> Position >> Linear Encoder >> Initial Position

**Data Type:** float64

**Description:** Specifies the position of the encoder when the measurement begins. This value is in the units you specify with Units.

You can get/set/reset this property using:

- `DAQmxGetCILinEncoderInitialPos`
- `DAQmxSetCILinEncoderInitialPos`
- `DAQmxResetCILinEncoderInitialPos`
Counter Input >> Position >> Decoding Type

Data Type: int32

Description: Specifies how to count and interpret the pulses the encoder generates on signal A and signal B. DAQmx_Val_X1, DAQmx_Val_X2, and DAQmx_Val_X4 are valid for quadrature encoders only. DAQmx_Val_TwoPulseCounting is valid for two-pulse encoders only.

X2 and X4 decoding are more sensitive to smaller changes in position than X1 encoding, with X4 being the most sensitive. However, more sensitive decoding is more likely to produce erroneous measurements if vibration exists in the encoder or other noise exists in the signals.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_X1</th>
<th>10090</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_X2</td>
<td>10091</td>
<td>Count the rising and falling edges of signal A.</td>
</tr>
<tr>
<td>DAQmx_Val_X4</td>
<td>10092</td>
<td>Count the rising and falling edges of signal A and signal B.</td>
</tr>
<tr>
<td>DAQmx_Val_TwoPulseCounting</td>
<td>10313</td>
<td>Increment the count on rising edges of signal A. Decrement the count on rising pulses of signal B.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetCIEncoderDecodingType
DAQmxSetCIEncoderDecodingType
DAQmxResetCIEncoderDecodingType
**Counter Input >> Position >> A Input >> Terminal**

**Data Type:** char*

**Description:** Specifies the terminal to which signal A is connected.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderAInputTerm`
- `DAQmxSetCIEncoderAInputTerm`
- `DAQmxResetCIEncoderAInputTerm`
Counter Input >> Position >> A Input >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderAInputDigFltrEnable`
- `DAQmxSetCIEncoderAInputDigFltrEnable`
- `DAQmxResetCIEncoderAInputDigFltrEnable`
**Counter Input >> Position >> A Input >> Digital Filter >> Minimum Pulse Width**

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderAInputDigFtrMinPulseWidth`
- `DAQmxSetCIEncoderAInputDigFtrMinPulseWidth`
- `DAQmxResetCIEncoderAInputDigFtrMinPulseWidth`
Counter Input >> Position >> A Input >> Digital Filter >> Timebase >> Source

**Data Type:**  char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- DAQmxGetCIEncoderAInputDigFltrTimebaseSrc
- DAQmxSetCIEncoderAInputDigFltrTimebaseSrc
- DAQmxResetCIEncoderAInputDigFltrTimebaseSrc
Counter Input >> Position >> A Input >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- DAQmxGetCIEncoderAInputDigFltrTimebaseRate
- DAQmxSetCIEncoderAInputDigFltrTimebaseRate
- DAQmxResetCIEncoderAInputDigFltrTimebaseRate
Counter Input >> Position >> A Input >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetCIEncoderAInputDigSyncEnable
- DAQmxSetCIEncoderAInputDigSyncEnable
- DAQmxResetCIEncoderAInputDigSyncEnable
Counter Input >> Position >> B Input >> Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which signal B is connected.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderBInputTerm`
- `DAQmxSetCIEncoderBInputTerm`
- `DAQmxResetCIEncoderBInputTerm`
Counter Input >> Position >> B Input >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderBInputDigFiltEnable`  
- `DAQmxSetCIEncoderBInputDigFiltEnable`  
- `DAQmxResetCIEncoderBInputDigFiltEnable`
Counter Input >> Position >> B Input >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- DAQmxGetCIEncoderBInputDigFtrMinPulseWidth
- DAQmxSetCIEncoderBInputDigFtrMinPulseWidth
- DAQmxResetCIEncoderBInputDigFtrMinPulseWidth
**Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Source**

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderBInputDigFltrTimebaseSrc`
- `DAQmxSetCIEncoderBInputDigFltrTimebaseSrc`
- `DAQmxResetCIEncoderBInputDigFltrTimebaseSrc`
Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderBInputDigFltrTimebaseRate`
- `DAQmxSetCIEncoderBInputDigFltrTimebaseRate`
- `DAQmxResetCIEncoderBInputDigFltrTimebaseRate`
Counter Input >> Position >> B Input >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderBInputDigSyncEnable`
- `DAQmxSetCIEncoderBInputDigSyncEnable`
- `DAQmxResetCIEncoderBInputDigSyncEnable`
**Counter Input >> Position >> Z Input >> Terminal**

**Data Type:** char*

**Description:** Specifies the terminal to which signal Z is connected.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderZInputTerm`
- `DAQmxSetCIEncoderZInputTerm`
- `DAQmxResetCIEncoderZInputTerm`
Counter Input >> Position >> Z Input >> Digital Filter >> Enable

Data Type: bool32
Description: Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

DAQmxGetCIEncoderZInputDigFtrEnable
DAQmxSetCIEncoderZInputDigFtrEnable
DAQmxResetCIEncoderZInputDigFtrEnable
Countert Input >> Position >> Z Input >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderZInputDigFtrMinPulseWidth`
- `DAQmxSetCIEncoderZInputDigFtrMinPulseWidth`
- `DAQmxResetCIEncoderZInputDigFtrMinPulseWidth`
Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderZInputDigFltrTimebaseSrc`
- `DAQmxSetCIEncoderZInputDigFltrTimebaseSrc`
- `DAQmxResetCIEncoderZInputDigFltrTimebaseSrc`
**Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Rate**

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderZInputDigFltrTimebaseRate`
- `DAQmxSetCIEncoderZInputDigFltrTimebaseRate`
- `DAQmxResetCIEncoderZInputDigFltrTimebaseRate`
Counter Input >> Position >> Z Input >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetCIEncoderZInputDigSyncEnable
- DAQmxSetCIEncoderZInputDigSyncEnable
- DAQmxResetCIEncoderZInputDigSyncEnable
Counter Input >> Position >> Z Index Enable

**Data Type:** bool32

**Description:** Specifies whether to use Z indexing for the channel.

You can get/set/reset this property using:

- DAQmxGetCIEncoderZIndexEnable
- DAQmxSetCIEncoderZIndexEnable
- DAQmxResetCIEncoderZIndexEnable
Counter Input >> Position >> Z Index Value

**Data Type:** float64

**Description:** 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 Z Index Phase. Specify this value in the units of the measurement.

You can get/set/reset this property using:

- `DAQmxGetCIEncoderZIndexVal`
- `DAQmxSetCIEncoderZIndexVal`
- `DAQmxResetCIEncoderZIndexVal`
**Counter Input >> Position >> Z Index Phase**

**Data Type:** int32

**Description:** 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 DAQmx_Val_AHighBHigh.

When signal Z transitions to high and how long it stays high varies from encoder to encoder. Refer to the documentation for the encoder to determine the timing of signal Z with respect to signal A and signal B.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AHighBHigh</td>
<td>10040</td>
<td>Reset the measurement when signal A and signal B are high.</td>
</tr>
<tr>
<td>DAQmx_Val_AHighBLow</td>
<td>10041</td>
<td>Reset the measurement when signal A is high and signal B is low.</td>
</tr>
<tr>
<td>DAQmx_Val_ALowBHigh</td>
<td>10042</td>
<td>Reset the measurement when signal A is low and signal B high.</td>
</tr>
<tr>
<td>DAQmx_Val_ALowBLow</td>
<td>10043</td>
<td>Reset the measurement when signal A and signal B are low.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIEncoderZIndexPhase
- DAQmxSetCIEncoderZIndexPhase
- DAQmxResetCIEncoderZIndexPhase
Counter Input >> Pulse Width >> Units

Data Type: int32
Description: Specifies the units to use to return pulse width measurements.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Seconds</th>
<th>10364</th>
<th>Seconds.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val.Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIPulseWidthUnits
- DAQmxSetCIPulseWidthUnits
- DAQmxResetCIPulseWidthUnits
Counter Input >> Pulse Width >> Input Terminal

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to measure.

You can get/set/reset this property using:

- DAQmxGetCIPulseWidthTerm
- DAQmxSetCIPulseWidthTerm
- DAQmxResetCIPulseWidthTerm
Counter Input >> Pulse Width >> Starting Edge

**Data Type:** int32

**Description:** Specifies on which edge of the input signal to begin each pulse width measurement.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCIPulseWidthStartingEdge
- DAQmxSetCIPulseWidthStartingEdge
- DAQmxResetCIPulseWidthStartingEdge
**Counter Input >> Pulse Width >> Digital Filter >> Enable**

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetCIPulseWidthDigFltrEnable`
- `DAQmxSetCIPulseWidthDigFltrEnable`
- `DAQmxResetCIPulseWidthDigFltrEnable`
Counter Input >> Pulse Width >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCIPulseWidthDigFltrMinPulseWidth`
- `DAQmxSetCIPulseWidthDigFltrMinPulseWidth`
- `DAQmxResetCIPulseWidthDigFltrMinPulseWidth`
Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- DAQmxGetCIPulseWidthDigFltrTimebaseSrc
- DAQmxSetCIPulseWidthDigFltrTimebaseSrc
- DAQmxResetCIPulseWidthDigFltrTimebaseSrc
Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCIPulseWidthDigFltrTimebaseRate`
- `DAQmxSetCIPulseWidthDigFltrTimebaseRate`
- `DAQmxResetCIPulseWidthDigFltrTimebaseRate`
Counter Input >> Pulse Width >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetCIPulseWidthDigSyncEnable`
- `DAQmxSetCIPulseWidthDigSyncEnable`
- `DAQmxResetCIPulseWidthDigSyncEnable`
Counter Input >> Two Edge Separation >> Units

**Data Type:** int32

**Description:** Specifies the units to use to return two-edge separation measurements from the channel.

### Valid values

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Seconds</td>
<td>10364</td>
<td>Seconds.</td>
</tr>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCTTwoEdgeSepUnits`
- `DAQmxSetCTwoEdgeSepUnits`
- `DAQmxResetCTTwoEdgeSepUnits`
Counter Input >> Two Edge Separation >> First >> Input Terminal

**Data Type:** char*

**Description:** Specifies the source terminal of the digital signal that starts each measurement.

You can get/set/reset this property using:

- `DAQmxGetCTwoEdgeSepFirstTerm`
- `DAQmxSetCTwoEdgeSepFirstTerm`
- `DAQmxResetCTwoEdgeSepFirstTerm`
Counter Input >> Two Edge Separation >> First >> Edge

**Data Type:** int32

**Description:** Specifies on which edge of the first signal to start each measurement.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCTwoEdgeSepFirstEdge`
- `DAQmxSetCTwoEdgeSepFirstEdge`
- `DAQmxResetCTwoEdgeSepFirstEdge`
Counter Input >> Two Edge Separation >> First >> Digital Filter >> Enable

Data Type: bool32
Description: Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- DAQmxGetCTwoEdgeSepFirstDigFtrEnable
- DAQmxSetCTwoEdgeSepFirstDigFtrEnable
- DAQmxResetCTwoEdgeSepFirstDigFtrEnable
Counter Input >> Two Edge Separation >> First >> Digital Filter >> Minimum Pulse Width

Data Type: float64
Description: Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCTTwoEdgeSepFirstDigFtrMinPulseWidth`
- `DAQmxSetCTTwoEdgeSepFirstDigFtrMinPulseWidth`
- `DAQmxResetCTTwoEdgeSepFirstDigFtrMinPulseWidth`
Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCTwoEdgeSepFirstDigFtrTimebaseSrc`
- `DAQmxSetCTwoEdgeSepFirstDigFtrTimebaseSrc`
- `DAQmxResetCTwoEdgeSepFirstDigFtrTimebaseSrc`
**Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Rate**

**Data Type:**  float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCTTwoEdgeSepFirstDigFltrTimebaseRate`
- `DAQmxSetCTTwoEdgeSepFirstDigFltrTimebaseRate`
- `DAQmxResetCTTwoEdgeSepFirstDigFltrTimebaseRate`
**Counter Input >> Two Edge Separation >> First >> Digital Synchronization >> Enable**

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetCTwoEdgeSepFirstDigSyncEnable`
- `DAQmxSetCTwoEdgeSepFirstDigSyncEnable`
- `DAQmxResetCTwoEdgeSepFirstDigSyncEnable`
Counter Input >> Two Edge Separation >> Second >> Input Terminal

**Data Type:**  char*

**Description:** Specifies the source `terminal` of the digital signal that stops each measurement.

You can *get/set/reset* this property using:

- `DAQmxGetCITwoEdgeSepSecondTerm`
- `DAQmxSetCITwoEdgeSepSecondTerm`
- `DAQmxResetCITwoEdgeSepSecondTerm`
Counter Input >> Two Edge Separation >> Second >> Edge

**Data Type:** int32  
**Description:** Specifies on which edge of the second signal to stop each measurement.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCITwoEdgeSepSecondEdge
- DAQmxSetCITwoEdgeSepSecondEdge
- DAQmxResetCITwoEdgeSepSecondEdge
Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Enable

Data Type: bool32

Description: Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

DAQmxGetCTTwoEdgeSepSecondDigFtrEnable
DAQmxSetCTTwoEdgeSepSecondDigFtrEnable
DAQmxResetCTTwoEdgeSepSecondDigFtrEnable
Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCTTwoEdgeSepSecondDigFltrMinPulseWidth`
- `DAQmxSetCTwoEdgeSepSecondDigFitrMinPulseWidth`
- `DAQmxResetCTTwoEdgeSepSecondDigFitrMinPulseWidth`
Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- DAQmxGetCTTwoEdgeSepSecondDigFltrTimebaseSrc
- DAQmxSetCTTwoEdgeSepSecondDigFltrTimebaseSrc
- DAQmxResetCTTwoEdgeSepSecondDigFltrTimebaseSrc
Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCTTwoEdgeSepSecondDigFltrTimebaseRate`
- `DAQmxSetCTTwoEdgeSepSecondDigFltrTimebaseRate`
- `DAQmxResetCTTwoEdgeSepSecondDigFltrTimebaseRate`
Counter Input >> Two Edge Separation >> Second >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetCITwoEdgeSepSecondDigSyncEnable
- DAQmxSetCITwoEdgeSepSecondDigSyncEnable
- DAQmxResetCITwoEdgeSepSecondDigSyncEnable
Counter Input >> Semi-Period >> Units

Data Type: int32

Description: Specifies the units to use to return semi-period measurements.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Seconds</th>
<th>10364</th>
<th>Seconds.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCISemiPeriodUnits
- DAQmxSetCISemiPeriodUnits
- DAQmxResetCISemiPeriodUnits
Counter Input >> Semi-Period >> Input Terminal

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to measure.

You can get/set/reset this property using:

- DAQmxGetCISemiPeriodTerm
- DAQmxSetCISemiPeriodTerm
- DAQmxResetCISemiPeriodTerm
Counter Input >> Semi-Period >> Starting Edge

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCISemiPeriodStartingEdge
- DAQmxSetCISemiPeriodStartingEdge
- DAQmxResetCISemiPeriodStartingEdge
Counter Input >> Semi-Period >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetCISemiPeriodDigFltrEnable`
- `DAQmxSetCISemiPeriodDigFltrEnable`
- `DAQmxResetCISemiPeriodDigFltrEnable`
Counter Input >> Semi-Period >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCISemiPeriodDigFltrMinPulseWidth`
- `DAQmxSetCISemiPeriodDigFltrMinPulseWidth`
- `DAQmxResetCISemiPeriodDigFltrMinPulseWidth`
**Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Source**

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetCISemiPeriodDigFltrTimebaseSrc`
- `DAQmxSetCISemiPeriodDigFltrTimebaseSrc`
- `DAQmxResetCISemiPeriodDigFltrTimebaseSrc`
Data Type: float64

Description: Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCISemiPeriodDigFltrTimebaseRate`
- `DAQmxSetCISemiPeriodDigFltrTimebaseRate`
- `DAQmxResetCISemiPeriodDigFltrTimebaseRate`
Counter Input >> Semi-Period >> Digital Synchronization >> Enable

**Data Type**: bool32

**Description**: Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetCISemiPeriodDigSyncEnable`
- `DAQmxSetCISemiPeriodDigSyncEnable`
- `DAQmxResetCISemiPeriodDigSyncEnable`
Counter Input >> Timestamp >> Units

**Data Type:** int32

**Description:** Specifies the units to use to return timestamp measurements.

### Valid values

<table>
<thead>
<tr>
<th>Data Code</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Seconds</td>
<td>10364</td>
<td>Seconds.</td>
</tr>
<tr>
<td>DAQmx_Val_FromCustomScale</td>
<td>10065</td>
<td>Units a custom scale specifies. If you select this value, you must specify a custom scale name.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCITimestampUnits
- DAQmxSetCITimestampUnits
- DAQmxResetCITimestampUnits
Counter Input >> Timestamp >> Initial Seconds

**Data Type:** uInt32

**Description:** Specifies the number of seconds that elapsed since the beginning of the current year. This value is ignored if [Synchronization Method](#) is DAQmx_Val_IRIGB.

You can get/set/reset this property using:

- DAQmxGetCTTimestampInitialSeconds
- DAQmxSetCTTimestampInitialSeconds
- DAQmxResetCTTimestampInitialSeconds
Counter Input >> Timestamp >> GPS >> Synchronization Method

**Data Type:** int32  
**Description:** Specifies the method to use to synchronize the counter to a GPS receiver.

### Valid values

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_IRIGB</td>
<td>10070</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_PPS</td>
<td>10080</td>
<td>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 Initial Seconds.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>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 Initial Seconds.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCIGPSSyncMethod`
- `DAQmxSetCIGPSSyncMethod`
- `DAQmxResetCIGPSSyncMethod`
**Counter Input >> Timestamp >> GPS >> Synchronization Source**

**Data Type:** char*

**Description:** Specifies the terminal to which the GPS synchronization signal is connected.

You can get/set/reset this property using:

- DAQmxGetCIGPSSyncSrc
- DAQmxSetCIGPSSyncSrc
- DAQmxResetCIGPSSyncSrc
Counter Input >> General Properties >> Counter Timebase >> Source

**Data Type:** char*

**Description:** Specifies the terminal of the timebase to use for the counter.

Typically, NI-DAQmx uses one of the internal counter timebases when performing counter measurements. Use this property to specify an external timebase and produce custom measurement ranges that are not possible with the internal timebases.

You also can use this property to chain counters together. By using the output of a counter as the timebase of another counter, you can effectively widen a counter. For example, you can chain two 24 bit counters together to produce one 48 bit counter.

You can get/set/reset this property using:

- `DAQmxGetCICtrTimebaseSrc`
- `DAQmxSetCICtrTimebaseSrc`
- `DAQmxResetCICtrTimebaseSrc`
Counter Input >> General Properties >> Counter Timebase >> Rate

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetCICtrTimebaseRate
- DAQmxSetCICtrTimebaseRate
- DAQmxResetCICtrTimebaseRate
Counter Input >> General Properties >> Counter Timebase >> Active Edge

**Data Type:** int32

**Description:** Specifies whether a timebase cycle is from rising edge to rising edge or from falling edge to falling edge.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetCICtrTimebaseActiveEdge
- DAQmxSetCICtrTimebaseActiveEdge
- DAQmxResetCICtrTimebaseActiveEdge
Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- DAQmxGetCICtrTimebaseDigFltrEnable
- DAQmxSetCICtrTimebaseDigFltrEnable
- DAQmxResetCICtrTimebaseDigFltrEnable
Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCICtrTimebaseDigFltrMinPulseWidth`
- `DAQmxSetCICtrTimebaseDigFltrMinPulseWidth`
- `DAQmxResetCICtrTimebaseDigFltrMinPulseWidth`
Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- DAQmxGetCICtrTimebaseDigFltrTimebaseSrc
- DAQmxSetCICtrTimebaseDigFltrTimebaseSrc
- DAQmxResetCICtrTimebaseDigFltrTimebaseSrc
**Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Rate**

**Data Type:**  float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCICtrTimebaseDigFltrTimebaseRate`
- `DAQmxSetCICtrTimebaseDigFltrTimebaseRate`
- `DAQmxResetCICtrTimebaseDigFltrTimebaseRate`
Counter Input >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.  
This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetCICtrTimebaseDigSyncEnable
- DAQmxSetCICtrTimebaseDigSyncEnable
- DAQmxResetCICtrTimebaseDigSyncEnable
Counter Input >> General Properties >> More >> Count

**Data Type:**  uInt32  
**Description:** Indicates the current value of the count register.  
**Restrictions:** Not Settable

You can get this property using:  

[DAQmxGetCICount](doc)
Counter Input >> General Properties >> More >>
Output State

Data Type: int32

Description: Indicates the current state of the out terminal of the counter.

Restrictions: Not Settable

Valid values

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>High state.</td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Low state.</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetCIOutputState
Counter Input >> General Properties >> More >>
Terminal Count Reached

**Data Type:** bool32

**Description:** Indicates whether the counter rolled over. When you query this property, NI-DAQmx resets it to FALSE.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetCITCReached](#)
Counter Input >> General Properties >> More >> Counter Timebase Master Timebase Divisor

**Data Type:**  uInt32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetCICtrTimebaseMasterTimebaseDiv`
- `DAQmxSetCICtrTimebaseMasterTimebaseDiv`
- `DAQmxResetCICtrTimebaseMasterTimebaseDiv`
**Data Transfer Mechanism**

**Data Type:** int32  
**Description:** Specifies the data transfer mode for the channel.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_DMA</th>
<th>10054</th>
<th>Direct Memory Access. Data transfers take place independently from the application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Interrupts</td>
<td>10204</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_ProgrammedIO</td>
<td>10264</td>
<td>Data transfers take place when you call an NI-DAQmx Read function or an NI-DAQmx Write function.</td>
</tr>
<tr>
<td>DAQmx_Val_USBbulk</td>
<td>12590</td>
<td>Data transfers take place independently from the application using a USB bulk pipe.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCIDataXferMech`
- `DAQmxSetCIDataXferMech`
- `DAQmxResetCIDataXferMech`
Counter Input >> General Properties >> More >> Advanced >> Data Transfer and Memory >> Number Of Possibly Invalid Samples

**Data Type:** uInt32

**Description:** Indicates the number of samples that the device might have overwritten before it could transfer them to the buffer.

On certain devices during finite buffered time measurements, it is not possible to detect if the counter overwrites a value it read before the device could transfer the sample to the buffer. This uncertainty is present only when Data Transfer Mechanism is DAQmx_Val_DMA and occurs as the acquisition nears completion.

Once the acquisition completes, it is impossible to tell if the status value on the counter indicates an overwrite as a result of a true overwrite, or because the counter detected another edge of the input signal after the acquisition completed but before the counter could disarm. As a result of this behavior, higher frequency input signals are more likely to increase the number of possibly invalid samples.

To decrease the number of possibly invalid samples, read more often from the buffer. Reading from the buffer forces a check on the hardware status and ensures all data is valid up to the present point in the buffer. By default, NI-DAQmx checks the validity of data every fourth of the buffer. If you do not perform a read until after the device acquires all data, the value of this property is one-fourth of the buffer size or zero.

The value this property indicates is valid only while the task is in the running state. The value of this property while the task is in any other state is zero.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetCINumPossiblyInvalidSamps
Counter Input >> General Properties >> More >> Advanced >> Duplicate Count Prevention

**Data Type:** bool32

**Description:** Specifies whether to enable duplicate count prevention for the channel. Duplicate count prevention is enabled by default. Setting Prescaler disables duplicate count prevention unless you explicitly enable it.

You can get/set/reset this property using:

- DAQmxGetCIDupCountPrevent
- DAQmxSetCIDupCountPrevent
- DAQmxResetCIDupCountPrevent
**Counter Input >> General Properties >> More >> Advanced >> Prescaler**

**Data Type:** uInt32

**Description:** 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 Duplicate Count Prevention to TRUE.

You can get/set/reset this property using:

- DAQmxGetCIPrescaler
- DAQmxSetCIPrescaler
- DAQmxResetCIPrescaler
Counter Output >> Output Type

**Data Type:** int32

**Description:** Indicates how to define pulses generated on the channel.

**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Pulse_Time</td>
<td>10269</td>
<td>Generate pulses defined by the time the pulse is at a low state and the time the pulse is at a high state.</td>
</tr>
<tr>
<td>DAQmx_Val_Pulse_Freq</td>
<td>10119</td>
<td>Generate digital pulses defined by frequency and duty cycle.</td>
</tr>
<tr>
<td>DAQmx_Val_Pulse_Ticks</td>
<td>10268</td>
<td>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.</td>
</tr>
</tbody>
</table>

You can get this property using:

`DAQmxGetCOOutputType`
Counter Output >> Pulse >> Idle State

Data Type: int32

Description: Specifies the resting state of the output terminal.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_High</th>
<th>10192</th>
<th>High state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Low state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCOPulseIdleState`
- `DAQmxSetCOPulseIdleState`
- `DAQmxResetCOPulseIdleState`
Counter Output >> Pulse >> Output Terminal

Data Type: char*

Description: Specifies on which terminal to generate pulses.

You can get/set/reset this property using:

- DAQmxGetCOPulseTerm
- DAQmxSetCOPulseTerm
- DAQmxResetCOPulseTerm
Counter Output >> Pulse >> Time >> Units

Data Type: int32

Description: Specifies the units in which to define high and low pulse time.

Valid values

| DAQmx_Val_Seconds | 10364 | Seconds. |

You can get/set/reset this property using:

- DAQmxGetCOPulseTimeUnits
- DAQmxSetCOPulseTimeUnits
- DAQmxResetCOPulseTimeUnits
Counter Output >> Pulse >> Time >> High Time

**Data Type:** float64

**Description:** Specifies the amount of time that the pulse is at a high voltage. This value is in the units you specify with Units or when you create the channel.

You can get/set/reset this property using:

- DAQmxGetCOPulseHighTime
- DAQmxSetCOPulseHighTime
- DAQmxResetCOPulseHighTime
Counter Output >> Pulse >> Time >> Low Time

**Data Type:** float64

**Description:** Specifies the amount of time that the pulse is at a low voltage. This value is in the units you specify with Units or when you create the channel.

You can get/set/reset this property using:

- `DAQmxGetCOPulseLowTime`
- `DAQmxSetCOPulseLowTime`
- `DAQmxResetCOPulseLowTime`
Counter Output >> Pulse >> Time >> Initial Delay

**Data Type:** float64

**Description:** Specifies in seconds the amount of time to wait before generating the first pulse.

You can get/set/reset this property using:

- `DAQmxGetCOPulseTimeInitialDelay`
- `DAQmxSetCOPulseTimeInitialDelay`
- `DAQmxResetCOPulseTimeInitialDelay`
**Counter Output >> Pulse >> Frequency >> Duty Cycle**

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetCOPulseDutyCyc
- DAQmxSetCOPulseDutyCyc
- DAQmxResetCOPulseDutyCyc
Counter Output >> Pulse >> Frequency >> Units

**Data Type:** int32

**Description:** Specifies the units in which to define pulse frequency.

**Valid values**

| DAQmx_Val_Hz | 10373 | Hertz |

You can get/set/reset this property using:

- DAQmxGetCPulseFreqUnits
- DAQmxSetCPulseFreqUnits
- DAQmxResetCPulseFreqUnits
**Counter Output >> Pulse >> Frequency >> Frequency**

**Data Type:** float64

**Description:** Specifies the frequency of the pulses to generate. This value is in the units you specify with [Units](#) or when you create the channel.

You can get/set/reset this property using:

- `DAQmxGetCOPulseFreq`
- `DAQmxSetCOPulseFreq`
- `DAQmxResetCOPulseFreq`
Counter Output >> Pulse >> Frequency >> Initial Delay

**Data Type:**  float64

**Description:** Specifies in seconds the amount of time to wait before generating the first pulse.

You can get/set/reset this property using:

- `DAQmxGetCOPulseFreqInitialDelay`
- `DAQmxSetCOPulseFreqInitialDelay`
- `DAQmxResetCOPulseFreqInitialDelay`
Counter Output >> Pulse >> Ticks >> High Ticks

**Data Type:**  uInt32

**Description:**  Specifies the number of ticks the pulse is high.

You can get/set/reset this property using:

- `DAQmxGetCOPulseHighTicks`
- `DAQmxSetCOPulseHighTicks`
- `DAQmxResetCOPulseHighTicks`
**Counter Output >> Pulse >> Ticks >> Low Ticks**

**Data Type:** \(\text{uInt32}\)

**Description:** Specifies the number of ticks the pulse is low.

You can get/set/reset this property using:

- `DAQmxGetCOPulseLowTicks`
- `DAQmxSetCOPulseLowTicks`
- `DAQmxResetCOPulseLowTicks`
Counter Output >> Pulse >> Ticks >> Initial Delay

Data Type:  uInt32

Description:  Specifies the number of ticks to wait before generating the first pulse.

You can get/set/reset this property using:

- DAQmxGetCOPulseTicksInitialDelay
- DAQmxSetCOPulseTicksInitialDelay
- DAQmxResetCOPulseTicksInitialDelay
Counter Output >> General Properties >> Counter Timebase >> Source

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetCOCtrTimebaseSrc`
- `DAQmxSetCOCtrTimebaseSrc`
- `DAQmxResetCOCtrTimebaseSrc`
Counter Output >> General Properties >> Counter Timebase >> Rate

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetCOOutTimebaseRate
- DAQmxSetCOOutTimebaseRate
- DAQmxResetCOOutTimebaseRate
**Counter Output >> General Properties >> Counter Timebase >> Active Edge**

**Data Type:** int32

**Description:** Specifies whether a timebase cycle is from rising edge to rising edge or from falling edge to falling edge.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value 1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetCOCtrTimebaseActiveEdge`
- `DAQmxSetCOCtrTimebaseActiveEdge`
- `DAQmxResetCOCtrTimebaseActiveEdge`
Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- DAQmxGetCOCtrTimebaseDigFtrEnable
- DAQmxSetCOCtrTimebaseDigFtrEnable
- DAQmxResetCOCtrTimebaseDigFtrEnable
Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetCOCtrTimebaseDigFltrMinPulseWidth`
- `DAQmxSetCOCtrTimebaseDigFltrMinPulseWidth`
- `DAQmxResetCOCtrTimebaseDigFltrMinPulseWidth`
Data Type: char*

Description: Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

DAQmxGetCOCtrTimebaseDigFltrTimebaseSrc
DAQmxSetCOCtrTimebaseDigFltrTimebaseSrc
DAQmxResetCOCtrTimebaseDigFltrTimebaseSrc
**Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Rate**

**Data Type:**  float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetCOCtrTimebaseDigFltrTimebaseRate`
- `DAQmxSetCOCtrTimebaseDigFltrTimebaseRate`
- `DAQmxResetCOCtrTimebaseDigFltrTimebaseRate`
Counter Output >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetCOCtrTimebaseDigSyncEnable
- DAQmxSetCOCtrTimebaseDigSyncEnable
- DAQmxResetCOCtrTimebaseDigSyncEnable
Counter Output >> General Properties >> More >> Count

**Data Type:** uInt32

**Description:** Indicates the current value of the count register.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetCOCount](#)
Counter Output >> General Properties >> More >> Output State

Data Type: int32
Description: Indicates the current state of the output terminal of the counter.
Restrictions: Not Settable

Valid values

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>High state.</td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Low state.</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetCOOutputState
Counter Output >> General Properties >> More >>

Auto Increment Count

**Data Type:** uInt32

**Description:** Specifies a number of timebase ticks by which to increment each successive pulse.

When this value is greater than 0, NI-DAQmx generates progressively longer pulses until the count register rolls over. At that point, the generated pulses return to the original pulse width and grow progressively longer until the count register rolls over again.

Use this property to provide a clock to an analog input channel for equivalent time sampling (ETS). ETS is a data acquisition technique in which data on a periodic waveform with a frequency higher than the Nyquist frequency of the system is obtained by sampling the waveform at instants in time skewed in relation to the beginning of each period of the waveform.

You can get/set/reset this property using:

- DAQmxGetCOAutoIncrCnt
- DAQmxSetCOAutoIncrCnt
- DAQmxResetCOAutoIncrCnt
Counter Output >> General Properties >> More >>
Counter Timebase Master Timebase Divisor

**Data Type:** uInt32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetCOCtrTimebaseMasterTimebaseDiv`
- `DAQmxSetCOCtrTimebaseMasterTimebaseDiv`
- `DAQmxResetCOCtrTimebaseMasterTimebaseDiv`
Counter Output >> General Properties >> More >> Pulse Done

**Data Type:** bool32

**Description:** 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. When you query this property, NI-DAQmx resets it to FALSE.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetCOPulseDone
Counter Output >> General Properties >> More >> Advanced >> Constrained Generation Mode

Data Type: int32

Description: 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.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Unconstrained</th>
<th>14708</th>
<th>Counter has no restrictions on pulse generation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FixedHighFreq</td>
<td>14709</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_FixedLowFreq</td>
<td>14710</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_Fixed50PercentDutyCycle</td>
<td>14711</td>
<td>Pulse duty cycle must be 50 percent. The frequency can change while the task runs.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetCOConstrainedGenMode
DAQmxSetCOConstrainedGenMode
DAQmxResetCOConstrainedGenMode
Counter Output >> General Properties >> More >> Advanced >> Prescaler

**Data Type:** uInt32

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetCOPrescaler
- DAQmxSetCOPrescaler
- DAQmxResetCOPrescaler
Counter Output >> General Properties >> More >> Advanced >> Ready For New Value

**Data Type:** bool32  
**Description:** Indicates whether the counter is ready for new continuous pulse train values.  
**Restrictions:** Not Settable

You can get this property using:  
[DAQmxGetCORdyForNewVal](#)
## General Properties >> Channel Type

**Data Type:** int32  
**Description:** Indicates the type of the virtual channel.  
**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_AI</th>
<th>10100</th>
<th>Analog input channel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AO</td>
<td>10102</td>
<td>Analog output channel.</td>
</tr>
<tr>
<td>DAQmx_Val_DI</td>
<td>10151</td>
<td>Digital input channel.</td>
</tr>
<tr>
<td>DAQmx_Val_DO</td>
<td>10153</td>
<td>Digital output channel.</td>
</tr>
<tr>
<td>DAQmx_Val_CI</td>
<td>10131</td>
<td>Counter input channel.</td>
</tr>
<tr>
<td>DAQmx_Val_CO</td>
<td>10132</td>
<td>Counter output channel.</td>
</tr>
</tbody>
</table>

You can get this property using:

`DAQmxGetChanType`
General Properties >> Physical Channel Name

**Data Type:** char*

**Description:** Specifies the name of the physical channel upon which this virtual channel is based.

You can get/set this property using:

- `DAQmxGetPhysicalChanName`
- `DAQmxSetPhysicalChanName`
General Properties >> Description

**Data Type:** char*

**Description:** Specifies a user-defined description for the channel.

You can get/set/reset this property using:

- `DAQmxGetChanDescr`
- `DAQmxSetChanDescr`
- `DAQmxResetChanDescr`
General Properties >> Is Global

**Data Type:** bool32

**Description:** Indicates whether the channel is a global channel.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetChanIsGlobal](#)
Device Is Simulated

**Data Type:** bool32

**Description:** Indicates if the device is a simulated device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevIsSimulated](#)
### Identification >> Product Category

**Data Type:** int32

**Description:** Indicates the product category of the device. This category corresponds to the category displayed in MAX when creating NI-DAQmx simulated devices.

**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_MSeriesDAQ</td>
<td>14643</td>
<td>M Series DAQ.</td>
</tr>
<tr>
<td>DAQmx_Val_ESeriesDAQ</td>
<td>14642</td>
<td>E Series DAQ.</td>
</tr>
<tr>
<td>DAQmx_Val_SSeriesDAQ</td>
<td>14644</td>
<td>S Series DAQ.</td>
</tr>
<tr>
<td>DAQmx_Val_BSeriesDAQ</td>
<td>14662</td>
<td>B Series DAQ.</td>
</tr>
<tr>
<td>DAQmx_Val_SCSeriesDAQ</td>
<td>14645</td>
<td>SC Series DAQ.</td>
</tr>
<tr>
<td>DAQmx_Val_USBDAQ</td>
<td>14646</td>
<td>USB DAQ.</td>
</tr>
<tr>
<td>DAQmx_Val_AOSeries</td>
<td>14647</td>
<td>AO Series.</td>
</tr>
<tr>
<td>DAQmx_Val_DigitalIO</td>
<td>14648</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>DAQmx_Val_TIOSeries</td>
<td>14661</td>
<td>TIO Series.</td>
</tr>
<tr>
<td>DAQmx_Val_DynamicSignalAcquisition</td>
<td>14649</td>
<td>Dynamic Signal Acquisition.</td>
</tr>
<tr>
<td>DAQmx_Val_Switches</td>
<td>14650</td>
<td>Switches.</td>
</tr>
<tr>
<td>DAQmx_Val_CompactDAQChassis</td>
<td>14658</td>
<td>CompactDAQ chassis.</td>
</tr>
<tr>
<td>DAQmx_Val_CSeriesModule</td>
<td>14659</td>
<td>C Series I/O module.</td>
</tr>
<tr>
<td>DAQmx_Val_SCXIModule</td>
<td>14660</td>
<td>SCXI module.</td>
</tr>
<tr>
<td>DAQmx_Val_SCCConnectorBlock</td>
<td>14704</td>
<td>SCC Connector Block.</td>
</tr>
<tr>
<td>DAQmx_Val_SCCModule</td>
<td>14705</td>
<td>SCC Module.</td>
</tr>
<tr>
<td>DAQmx_Val_NIELVIS</td>
<td>14755</td>
<td>NI ELVIS.</td>
</tr>
<tr>
<td>DAQmx_Val_Unknown</td>
<td>12588</td>
<td>Unknown category.</td>
</tr>
</tbody>
</table>

You can get this property using:

`DAQmxGetDevProductCategory`
Identification >> Product Type

Data Type: char*

Description: Indicates the product name of the device.

Restrictions: Not Settable

You can get this property using:

DAQmxGetDevProductType
Identification >> Product Number

Data Type:  uInt32

Description:  Indicates the unique hardware identification number for the device.

Restrictions:  Not Settable

You can get this property using:

DAQmxGetDevProductNum
Identification >> Device Serial Number

**Data Type:**  uInt32

**Description:** Indicates the serial number of the device. This value is zero if the device does not have a serial number.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevSerialNum](#)
Chassis >> Module Device Names

**Data Type:** char*

**Description:** Indicates an array containing the names of the modules in the chassis.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetDevChassisModuleDevNames
Analog Triggering Supported

**Data Type:** bool32

**Description:** Indicates if the device supports analog triggering.

**Restrictions:** Not Settable

You can get this property using:

```
DAQmxGetDevAnlgTrigSupported
```
Digital Triggering Supported

**Data Type:** bool32

**Description:** Indicates if the device supports digital triggering.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevDigTrigSupported`
I/O Type >> Analog Input >> Physical Channels

Data Type: char*

Description: Indicates an array containing the names of the analog input physical channels available on the device.

Restrictions: Not Settable

You can get this property using:

DAQmxGetDevAIPhysicalChans
I/O Type >> Analog Input >> Timing >> Maximum Single Channel Rate

**Data Type:** float64  
**Description:** Indicates the maximum rate for an analog input task if the task contains only a single channel from this device.  
**Restrictions:** Not Settable

You can get this property using:  
[DAQmxGetDevAlMaxSingleChanRate](DAQmxGetDevAlMaxSingleChanRate)
I/O Type >> Analog Input >> Timing >> Maximum Multiple Channel Rate

**Data Type:** float64

**Description:** Indicates the maximum rate for an analog input task if the task contains multiple channels from this device. For multiplexed devices, divide this rate by the number of channels to determine the maximum sampling rate.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevAlMaxMultiChanRate`
I/O Type >> Analog Input >> Timing >> Minimum Rate

**Data Type:** float64

**Description:** 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.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevAlMinRate`
I/O Type >> Analog Input >> Timing >> Simultaneous Sampling Supported

**Data Type:** bool32

**Description:** Indicates if the device supports simultaneous sampling.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetDevAIISimultaneousSamplingSupported
I/O Type >> Analog Input >> Trigger >> Trigger Usage

Data Type: int32
Description: Indicates the triggers supported by this device for an analog input task.
Restrictions: Not Settable

Valid values

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Advance</td>
<td>1</td>
<td>Device supports advance triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Pause</td>
<td>2</td>
<td>Device supports pause triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Reference</td>
<td>4</td>
<td>Device supports reference triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Start</td>
<td>8</td>
<td>Device supports start triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Handshake</td>
<td>16</td>
<td>Device supports handshake triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_ArmStart</td>
<td>32</td>
<td>Device supports arm start triggers</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetDevAITrigUsage
I/O Type >> Analog Input >> Voltage >> Ranges

**Data Type:** float64*

**Description:** Indicates pairs of input voltage ranges supported by this device. Each pair consists of the low value, followed by the high value.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevAIVoltageRngs`
I/O Type >> Analog Input >> Voltage >> Internal Excitation >> Discrete Values

**Data Type:** float64*

**Description:** Indicates the set of discrete internal voltage excitation values supported by this device. If the device supports ranges of internal excitation values, use [Range Values](#) to determine supported excitation values.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAlVoltageIntExcitDiscreteVals](#)
I/O Type >> Analog Input >> Voltage >> Internal Excitation >> Range Values

- **Data Type:** `float64`*
- **Description:** 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 [Discrete Values](#) to determine the supported excitation values.
- **Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevAIVoltageIntExcitRangeVals`
I/O Type >> Analog Input >> Current >> Ranges

Data Type: float64*

Description: Indicates the pairs of current input ranges supported by this device. Each pair consists of the low value, followed by the high value.

Restrictions: Not Settable

You can get this property using:

DAQmxGetDevAlCurrentRngs
I/O Type >> Analog Input >> Current >> Internal Excitation >> Discrete Values

**Data Type:** float64*

**Description:** Indicates the set of discrete internal current excitation values supported by this device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevAICurrentIntExcitDiscreteVals`
I/O Type >> Analog Input >> Frequency >> Ranges

**Data Type:**  float64*

**Description:** Indicates the pairs of frequency input ranges supported by this device. Each pair consists of the low value, followed by the high value.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAiFreqRngs](http://analoginputfrequencyranges.html)
I/O Type >> Analog Input >> Gains

**Data Type:** float64*

**Description:** Indicates the input gain settings supported by this device.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetDevAIGains
I/O Type >> Analog Input >> Couplings

**Data Type:** int32

**Description:** Indicates the coupling types supported by this device.

**Restrictions:** Not Settable

### Valid values

| DAQmx_Val_Bit_CouplingTypes_AC | 1 | Device supports AC coupling |
| DAQmx_Val_Bit_CouplingTypes_DC | 2 | Device supports DC coupling |
| DAQmx_Val_Bit_CouplingTypes_Ground | 4 | Device supports ground coupling |
| DAQmx_Val_Bit_CouplingTypes_HFReject | 8 | Device supports High Frequency Reject coupling |
| DAQmx_Val_Bit_CouplingTypes_LFReject | 16 | Device supports Low Frequency Reject coupling |
| DAQmx_Val_Bit_CouplingTypes_NoiseReject | 32 | Device supports Noise Reject coupling |

You can get this property using:

[DAQmxGetDevAICouplings](https://winspc.ni.com)
I/O Type >> Analog Input >> Filter >> Analog Low Pass >> Cutoff Frequency >> Discrete Values

Data Type: float64*
Description: Indicates the set of discrete lowpass cutoff frequencies supported by this device. If the device supports ranges of lowpass cutoff frequencies, use Range Values to determine supported frequencies.
Restrictions: Not Settable

You can get this property using:
DAQmxGetDevAILowpassCutoffFreqDiscreteVals
I/O Type >> Analog Input >> Filter >> Analog Low Pass >> Cutoff Frequency >> Range Values

**Data Type:** float64*

**Description:** 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 [Discrete Values](#) to determine the supported frequencies.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAllLowpassCutoffFreqRangeVals](#)
I/O Type >> Analog Output >> Physical Channels

**Data Type:** char*

**Description:** Indicates an array containing the names of the analog output physical channels available on the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAOPhysicalChan](#)
I/O Type >> Analog Output >> Timing >> Sample Clock Supported

**Data Type:** bool32

**Description:** Indicates if the device supports the sample clock timing type for analog output tasks.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetDevAOSampClkSupported
I/O Type >> Analog Output >> Timing >> Maximum Rate

**Data Type:**  float64

**Description:** Indicates the maximum analog output rate of the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAOMaxRate](https://github.com/DAQmxGetDevAOMaxRate)
I/O Type >> Analog Output >> Timing >> Minimum Rate

**Data Type:** float64

**Description:** Indicates the minimum analog output rate of the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAOMinRate](#)
I/O Type >> Analog Output >> Trigger >> Trigger Usage

**Data Type:** int32

**Description:** Indicates the triggers supported by this device for analog output tasks.

**Restrictions:** Not Settable

### Valid values

| DAQmx_Val_Bit_TriggerUsageTypes_Advance | 1 | Device supports advance triggers |
| DAQmx_Val_Bit_TriggerUsageTypes_Pause   | 2 | Device supports pause triggers   |
| DAQmx_Val_Bit_TriggerUsageTypes_Reference | 4 | Device supports reference triggers |
| DAQmx_Val_Bit_TriggerUsageTypes_Start | 8 | Device supports start triggers |
| DAQmx_Val_Bit_TriggerUsageTypes_Handshake | 16 | Device supports handshake triggers |
| DAQmx_Val_Bit_TriggerUsageTypes_ArmStart | 32 | Device supports arm start triggers |

You can get this property using:

`DAQmxGetDevAOTrigUsage`
I/O Type >> Analog Output >> Voltage >> Ranges

**Data Type**: float64*

**Description**: Indicates pairs of output voltage ranges supported by this device. Each pair consists of the low value, followed by the high value.

**Restrictions**: Not Settable

You can get this property using:

DAQmxGetDevAOVoltageRngs
I/O Type >> Analog Output >> Current >> Ranges

**Data Type:** float64*

**Description:** Indicates pairs of output current ranges supported by this device. Each pair consists of the low value, followed by the high value.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAOCurrentRngs](#)
I/O Type >> Analog Output >> Gains

**Data Type:** float64*

**Description:** Indicates the output gain settings supported by this device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevAOGains](https://example.com/DAQmxGetDevAOGains)
I/O Type >> Digital Input >> Lines

**Data Type:** char*

**Description:** Indicates an array containing the names of the digital input lines available on the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevDILines`
I/O Type >> Digital Input >> Ports

**Data Type:** char*

**Description:** Indicates an array containing the names of the digital input ports available on the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevDIPorts`
I/O Type >> Digital Input >> Timing >> Maximum Rate

**Data Type:** float64  
**Description:** Indicates the maximum digital input rate of the device.  
**Restrictions:** Not Settable

You can get this property using:  
[DAQmxGetDevDIMaxRate](#)
I/O Type >> Digital Input >> Trigger >> Trigger Usage

**Data Type:** int32

**Description:** Indicates the triggers supported by this device for digital input tasks.

**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Advance</td>
<td>Device supports advance triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Pause</td>
<td>Device supports pause triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Reference</td>
<td>Device supports reference triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Start</td>
<td>Device supports start triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Handshake</td>
<td>Device supports handshake triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_ArmStart</td>
<td>Device supports arm start triggers</td>
</tr>
</tbody>
</table>

You can get this property using:

[DAQmxGetDevDITrigUsage](https://example.com/DAQmxGetDevDITrigUsage)
I/O Type >> Digital Output >> Lines

**Data Type:** char*

**Description:** Indicates an array containing the names of the digital output lines available on the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevDOLines`
I/O Type >> Digital Output >> Ports

**Data Type:** char*

**Description:** Indicates an array containing the names of the digital output ports available on the device.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetDevDOPorts
I/O Type >> Digital Output >> Timing >> Maximum Rate

**Data Type:** float64

**Description:** Indicates the maximum digital output rate of the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevDOMaxRate](#)
**I/O Type >> Digital Output >> Trigger >> Trigger Usage**

**Data Type:** int32  
**Description:** Indicates the triggers supported by this device for digital output tasks.  
**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Advance</td>
<td>1 Device supports advance triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Pause</td>
<td>2 Device supports pause triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Reference</td>
<td>4 Device supports reference triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Start</td>
<td>8 Device supports start triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Handshake</td>
<td>16 Device supports handshake triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_ArmStart</td>
<td>32 Device supports arm start triggers</td>
</tr>
</tbody>
</table>

You can get this property using:  
I/O Type >> Counter Input >> Physical Channels

**Data Type:** char*

**Description:** Indicates an array containing the names of the counter input physical channels available on the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevCIPhysicalChans](#)
I/O Type >> Counter Input >> Trigger >> Trigger Usage

**Data Type:** int32

**Description:** Indicates the triggers supported by this device for counter input tasks.

**Restrictions:** Not Settable

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Advance</td>
<td>1 Device supports advance triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Pause</td>
<td>2 Device supports pause triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Reference</td>
<td>4 Device supports reference triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Start</td>
<td>8 Device supports start triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Handshake</td>
<td>16 Device supports handshake triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_ArmStart</td>
<td>32 Device supports arm start triggers</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetDevCITrigUsage
Data Type: bool32
Description: Indicates if the device supports the sample clock timing type for counter input tasks.
Restrictions: Not Settable

You can get this property using:

DAQmxGetDevCISeqClkSupported
I/O Type >> Counter Input >> Maximum Size

Data Type: UINT32  
Description: Indicates in bits the size of the counters on the device.  
Restrictions: Not Settable

You can get this property using:

DAQmxGetDevCIMaxSize
I/O Type >> Counter Input >> Maximum Timebase

- **Data Type:** float64
- **Description:** Indicates in hertz the maximum counter timebase frequency.
- **Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevClMaxTimebase`
I/O Type >> Counter Output >> Physical Channels

**Data Type:** char*

**Description:** Indicates an array containing the names of the counter output physical channels available on the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevCOPhysicalChans](https://examples.ni.com/nidev/doc/devext/sdk/reference/daqmxgetdevcophysicalchans)
**I/O Type >> Counter Output >> Trigger >> Trigger Usage**

**Data Type:** int32  
**Description:** Indicates the triggers supported by this device for counter output tasks.  
**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Advance</td>
<td>1 Device supports advance triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Pause</td>
<td>2 Device supports pause triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Reference</td>
<td>4 Device supports reference triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Start</td>
<td>8 Device supports start triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_Handshake</td>
<td>16 Device supports handshake triggers</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TriggerUsageTypes_ArmStart</td>
<td>32 Device supports arm start triggers</td>
</tr>
</tbody>
</table>

You can get this property using:  
`DAQmxGetDevCOTrigUsage`
I/O Type >> Counter Output >> Maximum Size

Data Type:  uInt32
Description:  Indicates in bits the size of the counters on the device.
Restrictions:  Not Settable

You can get this property using:

DAQmxGetDevCmaxSize
I/O Type >> Counter Output >> Maximum Timebase

- **Data Type:** float64
- **Description:** Indicates in hertz the maximum counter timebase frequency.
- **Restrictions:** Not Settable

You can get this property using:

DAQmxGetDevCOMaxTimebase
Location >> Bus Type

Data Type: int32
Description: Indicates the bus type of the device.
Restrictions: Not Settable

Valid values

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_PCI</td>
<td>12582</td>
<td>PCI</td>
</tr>
<tr>
<td>DAQmx_Val_PCIe</td>
<td>13612</td>
<td>PCI Express</td>
</tr>
<tr>
<td>DAQmx_Val_PXI</td>
<td>12583</td>
<td>PXI</td>
</tr>
<tr>
<td>DAQmx_Val_PXIe</td>
<td>14706</td>
<td>PXI Express</td>
</tr>
<tr>
<td>DAQmx_Val_SCXI</td>
<td>12584</td>
<td>SCXI</td>
</tr>
<tr>
<td>DAQmx_Val_SCC</td>
<td>14707</td>
<td>SCC</td>
</tr>
<tr>
<td>DAQmx_Val_PCCard</td>
<td>12585</td>
<td>PC Card/PCMCIA</td>
</tr>
<tr>
<td>DAQmx_Val_USB</td>
<td>12586</td>
<td>USB</td>
</tr>
<tr>
<td>DAQmx_Val_CompactDAQ</td>
<td>14637</td>
<td>CompactDAQ</td>
</tr>
<tr>
<td>DAQmx_Val_Unknown</td>
<td>12588</td>
<td>Unknown bus type</td>
</tr>
</tbody>
</table>

You can get this property using:

[DAQmxGetDevBusType](https://github.com/daqmx)
Bus >> Number of DMA Channels

**Data Type:** uint32

**Description:** Indicates the number of DMA channels on the device.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevNumDMAChans](https://example.com/DAQmxGetDevNumDMAChans)
Location >> PCI >> Bus Number

**Data Type:** uint32

**Description:** Indicates the PCI bus number of the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevPCIBusNum`
Location >> PCI >> Device Number

**Data Type:** uInt32

**Description:** Indicates the PCI slot number of the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevPCIDevNum`
Location >> PXI >> Chassis Number

**Data Type:**   UInt32  

**Description:** Indicates the PXI chassis number of the device, as identified in MAX.  

**Restrictions:**  Not Settable  

You can get this property using:  

[DAQmxGetDevPXIClassNum](#)
Location >> PXI >> Slot Number

**Data Type:**  uInt32  
**Description:**  Indicates the PXI slot number of the device. 
**Restrictions:**  Not Settable

You can get this property using:

DAQmxGetDevPXISlotNum
Location >> CompactDAQ >> Chassis Device Name

**Data Type:** char*

**Description:** Indicates the name of the CompactDAQ chassis that contains this module.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetDevCompactDAQChassisDevName`
Location >> CompactDAQ >> Slot Number

**Data Type:**  uInt32

**Description:** Indicates the slot number in which this module is located in the CompactDAQ chassis.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetDevCompactDAQSlotNum](#)
Terminals

Data Type: char*

Description: Indicates a list of all terminals on the device.

Restrictions: Not Settable

You can get this property using:

DAQmxGetDevTerminals
Clocks >> AI Convert Clock >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the AI Convert Clock.

You can get/set/reset this property using:

- `DAQmxGetExportedAIConvClkOutputTerm`
- `DAQmxSetExportedAIConvClkOutputTerm`
- `DAQmxResetExportedAIConvClkOutputTerm`
Clocks >> AI Convert Clock >> Pulse >> Polarity

**Data Type:** int32

**Description:** 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.

**Restrictions:** Not Settable

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>ValueCode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get this property using:

`DAQmxGetExportedAIConvClkPulsePolarity`
Clocks >> 10MHz Reference Clock >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the 10MHz Clock.

You can get/set/reset this property using:

- `DAQmxGetExported10MHzRefClkOutputTerm`
- `DAQmxSetExported10MHzRefClkOutputTerm`
- `DAQmxResetExported10MHzRefClkOutputTerm`
Clocks >> 20MHz Timebase >> Output Terminal

**Data Type:** char*

**Description:** Specifies the **terminal** to which to route the 20MHz Timebase.

You can get/set/reset this property using:

- DAQmxGetExported20MHzTimebaseOutputTerm
- DAQmxSetExported20MHzTimebaseOutputTerm
- DAQmxResetExported20MHzTimebaseOutputTerm
**Clocks >> Sample Clock >> Output Behavior**

**Data Type:** int32

**Description:** 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.

E Series devices might require many AI Convert Clock pulses to acquire one sample. Each pulse of the Sample Clock initiates the acquisition of one sample per channel in the task. Each sample per channel requires a pulse from the AI Convert Clock.

This property is valid for the AI Sample Clock only.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Pulse</th>
<th>10265</th>
<th>The exported Sample Clock pulses at the beginning of each sample.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Lvl</td>
<td>10210</td>
<td>The exported Sample Clock goes high at the beginning of the sample and goes low when the last AI Convert begins.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedSampClkOutputBehavior`
- `DAQmxSetExportedSampClkOutputBehavior`
- `DAQmxResetExportedSampClkOutputBehavior`
Clocks >> Sample Clock >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Sample Clock.

You can get/set/reset this property using:

- `DAQmxGetExportedSampClkOutputTerm`
- `DAQmxSetExportedSampClkOutputTerm`
- `DAQmxResetExportedSampClkOutputTerm`
Clocks >> Sample Clock >> Delay Offset

Data Type: float64

Description: 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.

You can get/set/reset this property using:

DAQmxGetExportedSampClkDelayOffset
DAQmxSetExportedSampClkDelayOffset
DAQmxResetExportedSampClkDelayOffset
Clocks >> Sample Clock >> Pulse >> Polarity

**Data Type:** int32

**Description:** Specifies the polarity of the exported Sample Clock if `Output Behavior` is `DAQmx_Val_Pulse`.

**Valid values**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DAQmx_Val_ActiveHigh</code></td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td><code>DAQmx_Val_ActiveLow</code></td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedSampClkPulsePolarity`
- `DAQmxSetExportedSampClkPulsePolarity`
- `DAQmxResetExportedSampClkPulsePolarity`
Clocks >> Sample Clock Timebase >> Output Terminal

**Data Type:** char

**Description:** Specifies the terminal to which to route the Sample Clock Timebase.

You can get/set/reset this property using:

- `DAQmxGetExportedSampClkTimebaseOutputTerm`
- `DAQmxSetExportedSampClkTimebaseOutputTerm`
- `DAQmxResetExportedSampClkTimebaseOutputTerm`
Clocks >> Divided Sample Clock Timebase >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Divided Sample Clock Timebase.

You can get/set/reset this property using:

- `DAQmxGetExportedDividedSampClkTimebaseOutputTerm`
- `DAQmxSetExportedDividedSampClkTimebaseOutputTerm`
- `DAQmxResetExportedDividedSampClkTimebaseOutputTerm`
Triggers >> Advance Trigger >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Advance Trigger.

You can get/set/reset this property using:

- `DAQmxGetExportedAdvTrigOutputTerm`
- `DAQmxSetExportedAdvTrigOutputTerm`
- `DAQmxResetExportedAdvTrigOutputTerm`
Triggers >> Advance Trigger >> Pulse >> Polarity

Data Type: int32
Description: Indicates the polarity of the exported Advance Trigger.
Restrictions: Not Settable

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_ActiveHigh</th>
<th>10095</th>
<th>High state is the active state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get this property using:

DAQmxGetExportedAdvTrigPulsePolarity
Triggers >> Advance Trigger >> Pulse >> Width Units

**Data Type:** int32

**Description:** Specifies the units of Width Value.

**Valid values**

| DAQmx_Val_Seconds   | 10364 | Seconds. |

You can get/set/reset this property using:

- DAQmxGetExportedAdvTrigPulseWidthUnits
- DAQmxSetExportedAdvTrigPulseWidthUnits
- DAQmxResetExportedAdvTrigPulseWidthUnits
**Triggers >> Advance Trigger >> Pulse >> Width Value**

**Data Type:** float64

**Description:** Specifies the width of an exported Advance Trigger pulse. Specify this value in the units you specify with Width Units.

You can get/set/reset this property using:

- DAQmxGetExportedAdvTrigPulseWidth
- DAQmxSetExportedAdvTrigPulseWidth
- DAQmxResetExportedAdvTrigPulseWidth
Triggers >> Pause Trigger >> Output Terminal

Data Type: char*

Description: Specifies the terminal to which to route the Pause Trigger.

You can get/set/reset this property using:

DAQmxGetExportedPauseTrigOutputTerm
DAQmxSetExportedPauseTrigOutputTerm
DAQmxResetExportedPauseTrigOutputTerm
Triggers >> Pause Trigger >> Level >> Active Level

**Data Type:** int32

**Description:** Specifies the active level of the exported Pause Trigger.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedPauseTrigLvlActiveLvl
- DAQmxSetExportedPauseTrigLvlActiveLvl
- DAQmxResetExportedPauseTrigLvlActiveLvl
Triggers >> Reference Trigger >> Output Terminal

Data Type: char*

Description: Specifies the terminal to which to route the Reference Trigger.

You can get/set/reset this property using:

-DAQmxGetExportedRefTrigOutputTerm
-DAQmxSetExportedRefTrigOutputTerm
-DAQmxResetExportedRefTrigOutputTerm
**Triggers >> Reference Trigger >> Pulse >> Polarity**

**Data Type:** int32

**Description:** Specifies the polarity of the exported Reference Trigger.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedRefTrigPulsePolarity
- DAQmxSetExportedRefTrigPulsePolarity
- DAQmxResetExportedRefTrigPulsePolarity
Triggers >> Start Trigger >> Output Terminal

**Data Type**: char*

**Description**: Specifies the terminal to which to route the Start Trigger.

You can get/set/reset this property using:

- DAQmxGetExportedStartTrigOutputTerm
- DAQmxSetExportedStartTrigOutputTerm
- DAQmxResetExportedStartTrigOutputTerm
Triggers >> Start Trigger >> Pulse >> Polarity

**Data Type:** int32

**Description:** Specifies the polarity of the exported Start Trigger.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_ActiveHigh</th>
<th>10095</th>
<th>High state is the active state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedStartTrigPulsePolarity
- DAQmxSetExportedStartTrigPulsePolarity
- DAQmxResetExportedStartTrigPulsePolarity
Events >> Advance Complete Event >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Advance Complete Event.

You can get/set/reset this property using:

- `DAQmxGetExportedAdvCmpltEventOutputTerm`
- `DAQmxSetExportedAdvCmpltEventOutputTerm`
- `DAQmxResetExportedAdvCmpltEventOutputTerm`
Events >> Advance Complete Event >> Delay Value

**Data Type:** float64

**Description:** Specifies the output signal delay in periods of the sample clock.

You can get/set/reset this property using:

- `DAQmxGetExportedAdvCmpltEventDelay`
- `DAQmxSetExportedAdvCmpltEventDelay`
- `DAQmxResetExportedAdvCmpltEventDelay`
**Events >> Advance Complete Event >> Pulse >> Polarity**

**Data Type:** int32

**Description:** Specifies the polarity of the exported Advance Complete Event.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedAdvCmpltEventPulsePolarity
- DAQmxSetExportedAdvCmpltEventPulsePolarity
- DAQmxResetExportedAdvCmpltEventPulsePolarity
Events >> Advance Complete Event >> Pulse >> Width Value

Data Type: float64

Description: Specifies the width of the exported Advance Complete Event pulse.

You can get/set/reset this property using:

DAQmxGetExportedAdvCmpltEventPulseWidth
DAQmxSetExportedAdvCmpltEventPulseWidth
DAQmxResetExportedAdvCmpltEventPulseWidth
Events >> AI Hold Complete Event >> Output Terminal

**Data Type:**  char*

**Description:** Specifies the terminal to which to route the AI Hold Complete Event.

You can get/set/reset this property using:

- DAQmxGetExportedAIHoldCmpltEventOutputTerm
- DAQmxSetExportedAIHoldCmpltEventOutputTerm
- DAQmxResetExportedAIHoldCmpltEventOutputTerm
Events >> AI Hold Complete Event >> Pulse >> Polarity

**Data Type:** int32

**Description:** Specifies the polarity of an exported AI Hold Complete Event pulse.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_ActiveHigh</th>
<th>10095</th>
<th>High state is the active state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedAIHoldCmpltEventPulsePolarity`
- `DAQmxSetExportedAIHoldCmpltEventPulsePolarity`
- `DAQmxResetExportedAIHoldCmpltEventPulsePolarity`
Events >> Change Detection Event >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Change Detection Event.

You can get/set/reset this property using:

- `DAQmxGetExportedChangeDetectEventOutputTerm`
- `DAQmxSetExportedChangeDetectEventOutputTerm`
- `DAQmxResetExportedChangeDetectEventOutputTerm`
Events >> Change Detection Event >> Pulse >> Polarity

Data Type: int32
Description: Specifies the polarity of an exported Change Detection Event pulse.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_ActiveHigh</th>
<th>10095</th>
<th>High state is the active state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedChangeDetectEventPulsePolarity
- DAQmxSetExportedChangeDetectEventPulsePolarity
- DAQmxResetExportedChangeDetectEventPulsePolarity
**Events >> Counter Output Event >> Output Terminal**

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Counter Output Event.

You can get/set/reset this property using:

- DAQmxGetExportedCtrOutEventOutputTerm
- DAQmxSetExportedCtrOutEventOutputTerm
- DAQmxResetExportedCtrOutEventOutputTerm
**Events >> Counter Output Event >> Output Behavior**

**Data Type:** int32

**Description:** Specifies whether the exported Counter Output Event pulses or changes from one state to the other when the counter reaches terminal count.

Upon reaching terminal count, the counter can issue a pulse. Use **Polarity** to select a high or low pulse.

Upon reaching terminal count, the output terminal of the counter can change state, or toggle. For example, if the terminal is initially at a low state, it changes to high state and stays at the high state until the next terminal count. The terminal then changes to low state. Use **Idle State** to select the initial state of the terminal.

When counting up, a counter reaches terminal count when it reaches the maximum value ($2^{24} - 1$ for a 24-bit counter). When counting down, a counter reaches terminal count when it reaches 0.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Pulse</th>
<th>10265</th>
<th>Send a pulse to the terminal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Toggle</td>
<td>10307</td>
<td>Toggle the state of the terminal from low to high or from high to low.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedCtrOutEventOutputBehavior`
- `DAQmxSetExportedCtrOutEventOutputBehavior`
- `DAQmxResetExportedCtrOutEventOutputBehavior`
Events >> Counter Output Event >> Pulse >> Polarity

**Data Type:** int32

**Description:** Specifies the polarity of the pulses at the output terminal of the counter when Output Behavior is DAQmx_Val_Pulse. NI-DAQmx ignores this property if Output Behavior is DAQmx_Val_Toggle.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_ActiveHigh</th>
<th>10095</th>
<th>High state is the active state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedCtrOutEventPulsePolarity`
- `DAQmxSetExportedCtrOutEventPulsePolarity`
- `DAQmxResetExportedCtrOutEventPulsePolarity`
Events >> Counter Output Event >> Toggle >> Idle State

**Data Type:** int32

**Description:** Specifies the initial state of the output terminal of the counter when Output Behavior is DAQmx_Val_Toggle. The terminal enters this state when NI-DAQmx commits the task.

The initial state of the terminal affects whether the first toggle is from low state to high state or from high state to low state.

NI-DAQmx ignores this property if Output Behavior is DAQmx_Val_Pulse.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_High</th>
<th>10192</th>
<th>High state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Low state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedCtrOutEventToggleIdleState
- DAQmxSetExportedCtrOutEventToggleIdleState
- DAQmxResetExportedCtrOutEventToggleIdleState
Events >> Handshake Event >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Handshake Event.

You can get/set/reset this property using:

- `DAQmxGetExportedHshkEventOutputTerm`
- `DAQmxSetExportedHshkEventOutputTerm`
- `DAQmxResetExportedHshkEventOutputTerm`
Events >> Handshake Event >> Output Behavior

**Data Type:** int32

**Description:** Specifies the output behavior of the Handshake Event.

**Valid values**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Interlocked</td>
<td>12549</td>
<td>Handshake Event deasserts after the Handshake Trigger asserts, plus the amount of time specified with <a href="#">Deassert Delay Value</a>.</td>
</tr>
<tr>
<td>DAQmx_Val_Pulse</td>
<td>10265</td>
<td>Handshake Event pulses with the pulse width specified in <a href="#">Width Value</a>.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedHshkEventOutputBehavior
- DAQmxSetExportedHshkEventOutputBehavior
- DAQmxResetExportedHshkEventOutputBehavior
Events >> Handshake Event >> Delay Value

**Data Type:** float64

**Description:** Specifies the number of seconds to delay after the Handshake Trigger deasserts before asserting the Handshake Event.

You can get/set/reset this property using:

- `DAQmxGetExportedHshkEventDelay`
- `DAQmxSetExportedHshkEventDelay`
- `DAQmxResetExportedHshkEventDelay`
Events >> Handshake Event >> Interlocked >> Asserted Level

**Data Type:** int32

**Description:** Specifies the asserted level of the exported Handshake Event if Output Behavior is DAQmx_Val_Interlocked.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_High</th>
<th>10192</th>
<th>High state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Low state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedHshkEventInterlockedAssertedLvl`
- `DAQmxSetExportedHshkEventInterlockedAssertedLvl`
- `DAQmxResetExportedHshkEventInterlockedAssertedLvl`
Events >> Handshake Event >> Interlocked >> Assert on Start

**Data Type:** bool32

**Description:** Specifies to assert the Handshake Event when the task starts if Output Behavior is DAQmx_Val_Interlocked.

You can get/set/reset this property using:

- `DAQmxGetExportedHshkEventInterlockedAssertOnStart`
- `DAQmxSetExportedHshkEventInterlockedAssertOnStart`
- `DAQmxResetExportedHshkEventInterlockedAssertOnStart`
Events >> Handshake Event >> Interlocked >> Deassert Delay Value

**Data Type:** float64

**Description:** Specifies in seconds the amount of time to wait after the Handshake Trigger asserts before deasserting the Handshake Event if **Output Behavior** is DAQmx_Val_Interlocked.

You can get/set/reset this property using:

- `DAQmxGetExportedHshkEventInterlockedDeassertDelay`
- `DAQmxSetExportedHshkEventInterlockedDeassertDelay`
- `DAQmxResetExportedHshkEventInterlockedDeassertDelay`
Events >> Handshake Event >> Pulse >> Polarity

**Data Type:** int32

**Description:** Specifies the polarity of the exported Handshake Event if Output Behavior is DAQmx_Val_Pulse.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_ActiveHigh</th>
<th>10095</th>
<th>High state is the active state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedHshkEventPulsePolarity
- DAQmxSetExportedHshkEventPulsePolarity
- DAQmxResetExportedHshkEventPulsePolarity
**Events >> Handshake Event >> Pulse >> Width Value**

**Data Type:** float64

**Description:** Specifies in seconds the pulse width of the exported Handshake Event if Output Behavior is DAQmx_Val_Pulse.

You can get/set/reset this property using:

- `DAQmxGetExportedHshkEventPulseWidth`
- `DAQmxSetExportedHshkEventPulseWidth`
- `DAQmxResetExportedHshkEventPulseWidth`
Events >> Ready For Transfer Event >> Output Terminal

**Data Type:** char*

**Description:** Specifies the terminal to which to route the Ready for Transfer Event.

You can get/set/reset this property using:

- `DAQmxGetExportedRdyForXferEventOutputTerm`
- `DAQmxSetExportedRdyForXferEventOutputTerm`
- `DAQmxResetExportedRdyForXferEventOutputTerm`
Events >> Ready For Transfer Event >> Level >> Active Level

**Data Type:** int32

**Description:** Specifies the active level of the exported Ready for Transfer Event.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedRdyForXferEventLvlActiveLvl`
- `DAQmxSetExportedRdyForXferEventLvlActiveLvl`
- `DAQmxResetExportedRdyForXferEventLvlActiveLvl`
Events >> Ready For Transfer Event >> Deassert Condition

Data Type:  int32
Description:  Specifies when the ready for transfer event deasserts.

Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_OnbrdMemMoreThanHalfFull</td>
<td>Deassert the signal when more than half of the onboard memory of the device fills.</td>
</tr>
<tr>
<td>DAQmx_Val_OnbrdMemFull</td>
<td>Deassert the signal when the onboard memory fills.</td>
</tr>
<tr>
<td>DAQmx_Val_OnbrdMemCustomThreshold</td>
<td>Deassert the signal when the amount of space available in the onboard memory is below the value specified with Deassert Condition Custom Threshold.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedRdyForXferEventDeassertCond
- DAQmxSetExportedRdyForXferEventDeassertCond
- DAQmxResetExportedRdyForXferEventDeassertCond
Events >> Ready For Transfer Event >> Deassert Condition Custom Threshold

Data Type: uInt32

Description: 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. Deassert Condition must be DAQmx_Val_OnbrdMemCustomThreshold to use a custom threshold.

You can get/set/reset this property using:

- DAQmxGetExportedRdyForXferEventDeassertCondCustomThreshold
- DAQmxSetExportedRdyForXferEventDeassertCondCustomThreshold
- DAQmxResetExportedRdyForXferEventDeassertCondCustomThreshold
Events >> Data Active Event >> Output Terminal

Data Type: char*

Description: Specifies the terminal to which to export the Data Active Event.

You can get/set/reset this property using:

DAQmxGetExportedDataActiveEventOutputTerm
DAQmxSetExportedDataActiveEventOutputTerm
DAQmxResetExportedDataActiveEventOutputTerm
Events >> Data Active Event >> Level >> Active Level

**Data Type:** int32

**Description:** Specifies the polarity of the exported Data Active Event.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetExportedDataActiveEventLvlActiveLvl`
- `DAQmxSetExportedDataActiveEventLvlActiveLvl`
- `DAQmxResetExportedDataActiveEventLvlActiveLvl`
Events >> Ready For Start Event >> Output Terminal

Data Type: char*

Description: Specifies the terminal to which to route the Ready for Start Event.

You can get/set/reset this property using:

DAQmxGetExportedRdyForStartEventOutputTerm
DAQmxSetExportedRdyForStartEventOutputTerm
DAQmxResetExportedRdyForStartEventOutputTerm
Events >> Ready For Start Event >> Level >> Active Level

**Data Type:** int32

**Description:** Specifies the polarity of the exported Ready for Start Event.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_ActiveHigh</td>
<td>10095</td>
<td>High state is the active state.</td>
</tr>
<tr>
<td>DAQmx_Val_ActiveLow</td>
<td>10096</td>
<td>Low state is the active state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetExportedRdyForStartEventLvlActiveLvl
- DAQmxSetExportedRdyForStartEventLvlActiveLvl
- DAQmxResetExportedRdyForStartEventLvlActiveLvl
Events >> Synchronization Pulse Event >> Output Terminal

Data Type: char*
Description: Specifies the terminal to which to route the Synchronization Pulse Event.

You can get/set/reset this property using:

DAQmxGetExportedSyncPulseEventOutputTerm
DAQmxSetExportedSyncPulseEventOutputTerm
DAQmxResetExportedSyncPulseEventOutputTerm
Events >> Watchdog Timer Expired Event >> Output Terminal

**Data Type:**  char*

**Description:** Specifies the terminal to which to route the Watchdog Timer Expired Event.

You can get/set/reset this property using:

- DAQmxGetExportedWatchdogExpiredEventOutputTerm
- DAQmxSetExportedWatchdogExpiredEventOutputTerm
- DAQmxResetExportedWatchdogExpiredEventOutputTerm
Author

Data Type: char*

Description: Indicates the author of the global channel.

Restrictions: Not Settable

You can get this property using:

DAQmxGetPersistedChanAuthor
Allow Interactive Editing?

Data Type: bool32

Description: Indicates whether the global channel can be edited in the DAQ Assistant.

Restrictions: Not Settable

You can get this property using:

DAQmxGetPersistedChanAllowInteractiveEditing
Allow Interactive Deletion?

Data Type: bool32

Description: Indicates whether the global channel can be deleted through MAX.

Restrictions: Not Settable

You can get this property using:

`DAQmxGetPersistedChanAllowInteractiveDeletion`
### Author

**Data Type:** char*

**Description:** Indicates the author of the custom scale.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetPersistedScaleAuthor`
Allow Interactive Editing?

**Data Type:** bool32

**Description:** Indicates whether the custom scale can be edited in the DAQ Assistant.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPersistedScaleAllowInteractiveEditing](https://developers.ni.com/ NI Technical Support)
Allow Interactive Deletion?

**Data Type:** bool32

**Description:** Indicates whether the custom scale can be deleted through MAX.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPersistedScaleAllowInteractiveDeletion](#)
**Author**

**Data Type:** char*

**Description:** Indicates the author of the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPersistedTaskAuthor](#)
Allow Interactive Editing?

**Data Type:** bool32  
**Description:** Indicates whether the task can be edited in the DAQ Assistant.  
**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPersistedTaskAllowInteractiveEditing](https://...
Allow Interactive Deletion?

**Data Type:** bool32

**Description:** Indicates whether the task can be deleted through MAX.

**Restrictions:** Not Settable

You can get this property using:

DAOmxGetPersistedTaskAllowInteractiveDeletion
**Analog Input >> Input Configuration >> Terminal Configurations**

**Data Type:** int32  
**Description:** Indicates the list of terminal configurations supported by the channel.  
**Restrictions:** Not Settable

<table>
<thead>
<tr>
<th>Valid values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TermCfg_RSE</td>
<td>1 RSE terminal configuration</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TermCfg_NRSE</td>
<td>2 NRSE terminal configuration</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TermCfg_Diff</td>
<td>4 Differential terminal configuration</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TermCfg_PseudoDIFF</td>
<td>8 Pseudodifferential terminal configuration</td>
</tr>
</tbody>
</table>

You can get this property using:

```
DAQmxGetPhysicalChanAITermCfgs
```
Analog Output >> Output Configuration >> Terminal Configurations

**Data Type:** int32  
**Description:** Indicates the list of terminal configurations supported by the channel.  
**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Bit_TermCfg_RSE</td>
<td>1 RSE terminal configuration</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TermCfg_NRSE</td>
<td>2 NRSE terminal configuration</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TermCfg_Diff</td>
<td>4 Differential terminal configuration</td>
</tr>
<tr>
<td>DAQmx_Val_Bit_TermCfg_PseudoDIFF</td>
<td>8 Pseudodifferential terminal configuration</td>
</tr>
</tbody>
</table>

You can get this property using:  
[DAQmxGetPhysicalChanAOTermCfgs](https://www.ni.com)
Analog Output >> Advanced >> Manual Control >> Enable

**Data Type:** bool32

**Description:** 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-DAQmx.

You can get/set/reset this property using:

- `DAQmxGetPhysicalChanAOManualControlEnable`
- `DAQmxSetPhysicalChanAOManualControlEnable`
- `DAQmxResetPhysicalChanAOManualControlEnable`
Analog Output >> Advanced >> Manual Control >> Amplitude

**Data Type:** float64

**Description:** Indicates the current value of the front panel amplitude control for the physical channel in volts.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPhysicalChanAOManualControlAmplitude](#)
Analog Output >> Advanced >> Manual Control >> Frequency

**Data Type:** float64

**Description:** Indicates the current value of the front panel frequency control for the physical channel in hertz.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPhysicalChanAOManualControlFreq](https://www.ni.com)
**Digital Input >> Port Width**

**Data Type:** uInt32  
**Description:** Indicates in bits the width of digital input port.  
**Restrictions:** Not Settable

You can get this property using:  
Digital Input >> Timing >> Sample Clock Supported

**Data Type:** bool32

**Description:** Indicates if the sample clock timing type is supported for the digital input physical channel.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetPhysicalChanDISampClkSupported`
Digital Input >> Timing >> Change Detection Supported

**Data Type:** bool32

**Description:** Indicates if the change detection timing type is supported for the digital input physical channel.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPhysicalChanDIChangeDetectSupported](#)
**Digital Output >> Port Width**

**Data Type:** uInt32

**Description:** Indicates in bits the width of digital output port.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetPhysicalChanDOPortWidth
Digital Output >> Timing >> Sample Clock Supported

**Data Type:** bool32

**Description:** Indicates if the sample clock timing type is supported for the digital output physical channel.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPhysicalChanDOSampClkSupported](https://example.com/daqmxgetphysicalchandosampclksupported)
TEDS >> ManufacturerID

**Data Type:**  uInt32

**Description:** Indicates the manufacturer ID of the sensor.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetPhysicalChanTEDSMfgID](http://example.com/DAQmxGetPhysicalChanTEDSMfgID)
TEDS >> Model Number

**Data Type:** uInt32  
**Description:** Indicates the model number of the sensor.  
**Restrictions:** Not Settable

You can get this property using:  

`DAQmxGetPhysicalChanTEDSModelNum`
TEDS >> Serial Number

Data Type:  uInt32
Description:  Indicates the serial number of the sensor.
Restrictions:  Not Settable

You can get this property using:

DAQmxGetPhysicalChanTEDSSerialNum
TEDS >> Version Number

**Data Type:** Uint32  
**Description:** Indicates the version number of the sensor.  
**Restrictions:** Not Settable

You can get this property using:

DAQmxGetPhysicalChanTEDSVersionNum
TEDS >> Version Letter

Data Type: char*
Description: Indicates the version letter of the sensor.
Restrictions: Not Settable

You can get this property using:

DAQmxGetPhysicalChanTEDSVersionLetter
TEDS >> BitStream

**Data Type:** `uint8*`

**Description:** Indicates the TEDS binary bitstream without checksums.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetPhysicalChanTEDSBitStream`
TEDS >> TemplateIDs

**Data Type:**  uInt32*

**Description:** Indicates the IDs of the templates in the bitstream in BitStream.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetPhysicalChanTEDSTemplateIDs
RelativeTo

**Data Type:** int32

**Description:** 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 DAQmx_Val_CurrReadPos unless you configure a Reference Trigger for the task. If you configure a Reference Trigger, the default value is DAQmx_Val_FirstPretrigSamp.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FirstSample</td>
<td>10424</td>
<td>Start reading samples relative to the first sample acquired.</td>
</tr>
<tr>
<td>DAQmx_Val_CurrReadPos</td>
<td>10425</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_RefTrig</td>
<td>10426</td>
<td>Start reading samples relative to the first sample after the reference trigger occurred.</td>
</tr>
<tr>
<td>DAQmx_Val_FirstPretrigSamp</td>
<td>10427</td>
<td>Start reading samples relative to the first pretrigger sample. You specify the number of pretrigger samples to acquire when you configure a reference trigger.</td>
</tr>
<tr>
<td>DAQmx_Val_MostRecentSamp</td>
<td>10428</td>
<td>Start reading samples relative to the next sample acquired. For example, use this value and set <strong>Offset</strong> to -1 to read the last sample acquired.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetReadRelativeTo
- DAQmxSetReadRelativeTo
- DAQmxResetReadRelativeTo
Offset

**Data Type:** int32

**Description:** Specifies an offset in samples per channel at which to begin a read operation. This offset is relative to the location you specify with `RelativeTo`.

You can get/set/reset this property using:

- `DAQmxGetReadOffset`
- `DAQmxSetReadOffset`
- `DAQmxResetReadOffset`
Channels to Read

**Data Type:** char*

**Description:** Specifies a subset of channels in the task from which to read.

You can get/set/reset this property using:

- `DAQmxGetReadChannelsToRead`
- `DAQmxSetReadChannelsToRead`
- `DAQmxResetReadChannelsToRead`
**Read All Available Samples**

**Data Type:** bool32

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetReadReadAllAvailSamp`
- `DAQmxSetReadReadAllAvailSamp`
- `DAQmxResetReadReadAllAvailSamp`
Auto Start

**Data Type:** bool32

**Description:** Specifies if an NI-DAQmx Read function automatically starts the task if you did not start the task explicitly by using DAQmxStartTask(). The default value is TRUE. When an NI-DAQmx Read function starts a finite acquisition task, it also stops the task after reading the last sample.

You can get/set/reset this property using:

- DAQmxGetReadAutoStart
- DAQmxSetReadAutoStart
- DAQmxResetReadAutoStart
**OverWrite Mode**

**Data Type:** int32  
**Description:** Specifies whether to overwrite samples in the buffer that you have not yet read.

### Valid values

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_OverwriteUnreadSamps</td>
<td>10252</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DAQmx_Val_DoNotOverwriteUnreadSamps</td>
<td>10159</td>
</tr>
</tbody>
</table>

**DAQmx_Val_OverwriteUnreadSamps** 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 **RelativeTo** to **DAQmx_Val_MostRecentSamp** and setting **Offset** to the appropriate number of samples.

**DAQmx_Val_DoNotOverwriteUnreadSamps** 10159  
The acquisition stops when it encounters a sample in the buffer that you have not read.

You can get/set/reset this property using:

- **DAQmxGetReadOverWrite**
- **DAQmxSetReadOverWrite**
- **DAQmxResetReadOverWrite**
**Status >> Current Read Position**

**Data Type:** uint64

**Description:** Indicates in samples per channel the current position in the buffer.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetReadCurrReadPos](#)
Status >> Available Samples Per Channel

**Data Type:** uInt32

**Description:** Indicates the number of samples available to read per channel. This value is the same for all channels in the task.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetReadAvailSampPerChan`
Status >> Total Samples Per Channel Acquired

**Data Type:** uInt64

**Description:** 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.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetReadTotalSampPerChanAcquired`


**Status >> Overcurrent >> Overcurrent Channels Exist**

**Data Type:** bool32

**Description:** 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 Channels. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetReadOvercurrentChansExist`
Status >> Overcurrent >> Overcurrent Channels

**Data Type:** char*

**Description:** Indicates the names of any virtual channels in the task for which an overcurrent condition has been detected. You must read Overcurrent Channels 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.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetReadOvercurrentChans](https://example.com/daqmxgetreadovercurrentchans)
Status >> Open Current Loop >> Open Current Loop Channels Exist

**Data Type:** bool32

**Description:** 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 Channels. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetReadOpenCurrentLoopChansExist
### Status >> Open Current Loop >> Open Current Loop Channels

**Data Type:** char*

**Description:** Indicates the names of any virtual channels in the task for which the device(s) detected an open current loop. You must read Open Current Loop Channels Exist before you read this property. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetReadOpenCurrentLoopChans`
Status >> Overload >> Overloaded Channels Exist

**Data Type:** bool32

**Description:** 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 Channels. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetReadOverloadedChansExist](#)
**Status >> Overload >> Overloaded Channels**

**Data Type:** char*

**Description:** Indicates the names of any overloaded virtual channels in the task. You must read Overloaded Channels Exist before you read this property. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetReadOverloadedChans
Status >> Advanced >> ChangeDetection >> Overflowed

**Data Type:** bool32

**Description:** Indicates if samples were missed because change detection events occurred faster than the device could handle them. Some devices detect overflows differently than others.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetReadChangeDetectHasOverflowed`
**Advanced >> Raw Data Width**

**Data Type:** uInt32

**Description:** Indicates in bytes the size of a raw sample from the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetReadRawDataWidth](#)
**Advanced >> Number of Channels**

**Data Type:**  uInt32

**Description:** Indicates the number of channels that an NI-DAQmx Read function 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](#).

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetReadNumChans](#)
Advanced >> Digital Input >> Number of Bytes Per Channel

**Data Type:** uInt32

**Description:** Indicates the number of bytes per channel that NI-DAQmx returns in a sample for line-based reads. If a channel has fewer lines than this number, the extra bytes are FALSE.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetReadDigitalLinesBytesPerChan](#)
Advanced >> Wait Mode

Data Type: int32

Description: Specifies how an NI-DAQmx Read function waits for samples to become available.

Valid values

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_WaitForInterrupt</td>
<td>12523</td>
</tr>
<tr>
<td>DAQmx_Val_Poll</td>
<td>12524</td>
</tr>
<tr>
<td>DAQmx_Val_Yield</td>
<td>12525</td>
</tr>
<tr>
<td>DAQmx_Val_Sleep</td>
<td>12547</td>
</tr>
</tbody>
</table>

- DAQmx_Val_WaitForInterrupt (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.
- DAQmx_Val_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.
- DAQmx_Val_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.
- DAQmx_Val_Sleep (12547): Check for available samples once per the amount of time specified in Sleep Time.

You can get/set/reset this property using:

DAQmxGetReadWaitMode
DAQmxSetReadWaitMode
DAQmxResetReadWaitMode
Advanced >> Sleep Time

**Data Type:** float64

**Description:** Specifies in seconds the amount of time to sleep after checking for available samples if [Wait Mode](#) is `DAQmx_Val_Sleep`.

You can get/set/reset this property using:

- `DAQmxGetReadSleepTime`
- `DAQmxSetReadSleepTime`
- `DAQmxResetReadSleepTime`
Convert Late Errors To Warnings

**Data Type:** bool32

**Description:** Specifies if DAQmxWaitForNextSampleClock() and an NI-DAQmx Read function convert late errors to warnings. NI-DAQmx returns no late warnings or errors until the number of warmup iterations you specify with **Number Of Warmup Iterations** execute.

You can get/set/reset this property using:

- DAQmxGetRealTimeConvLateErrorsToWarnings
- DAQmxSetRealTimeConvLateErrorsToWarnings
- DAQmxResetRealTimeConvLateErrorsToWarnings
Number Of Warmup Iterations

**Data Type:** uint32

**Description:** Specifies the number of loop iterations that must occur before `DAQmxWaitForNextSampleClock()` and an NI-DAQmx Read function return any late warnings or errors. The system needs a number of iterations to stabilize. During this period, a large amount of jitter occurs, potentially causing reads and writes to be late. The default number of warmup iterations is 100. Specify a larger number if needed to stabilize the system.

You can get/set/reset this property using:

- `DAQmxGetRealTimeNumOfWarmupIters`
- `DAQmxSetRealTimeNumOfWarmupIters`
- `DAQmxResetRealTimeNumOfWarmupIters`
Wait For Next Sample Clock Wait Mode

**Data Type:** int32

**Description:** Specifies how DAQmxWaitForNextSampleClock() waits for the next Sample Clock pulse.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_WaitForInterrupt</td>
<td>12523</td>
<td>Check for Sample Clock pulses when the system receives an interrupt service request. This mode is the most CPU efficient, but results in lower possible sampling rates.</td>
</tr>
<tr>
<td>DAQmx_Val_Poll</td>
<td>12524</td>
<td>Repeatedly check for Sample Clock pulses as fast as possible. This mode allows for the highest sampling rates at the expense of CPU efficiency.</td>
</tr>
<tr>
<td>DAQmx_Val_Yield</td>
<td>12525</td>
<td>Repeatedly check for Sample Clock pulses, but yield control to other threads after each check. This mode offers a balance between sampling rate and CPU efficiency.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetRealTimeWaitForNextSampClkWaitMode`
- `DAQmxSetRealTimeWaitForNextSampClkWaitMode`
- `DAQmxResetRealTimeWaitForNextSampClkWaitMode`
Report Missed Samples

Data Type: bool32

Description: Specifies whether an NI-DAQmx Read function returns lateness errors or warnings when it detects missed Sample Clock pulses. This setting does not affect DAQmxWaitForNextSampleClock(). Set this property to TRUE for applications that need to detect lateness without using DAQmxWaitForNextSampleClock().

You can get/set/reset this property using:

DAQmxGetRealTimeReportMissedSamp
DAQmxSetRealTimeReportMissedSamp
DAQmxResetRealTimeReportMissedSamp
Write Recovery Mode

**Data Type:** int32

**Description:** Specifies how NI-DAQmx attempts to recover after missing a Sample Clock pulse when performing counter writes.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_WaitForInterrupt</th>
<th>12523</th>
<th>Attempt to recover when the system receives an interrupt service request. This mode is the most CPU efficient and best suited for recovery at lower pulse train frequencies.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Poll</td>
<td>12524</td>
<td>Repeatedly attempt to recover as fast as possible. This mode has the highest probability of recovery success at the expense of CPU efficiency.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetValue
- DAQmxSetValue
- DAQmxResetValue
- DAQmxGetRealTimeWriteRecoveryMode
- DAQmxSetRealTimeWriteRecoveryMode
- DAQmxResetRealTimeWriteRecoveryMode
**Description**

**Data Type:** char*

**Description:** Specifies a description for the scale.

You can get/set this property using:

DAQmxGetScaleDescr
DAQmxSetScaleDescr
**Scaled Units**

**Data Type:** char*

**Description:** Specifies the units to use for scaled values. You can use an arbitrary string.

You can get/set this property using:

- `DAQmxGetScaleScaledUnits`
- `DAQmxSetScaleScaledUnits`
Pre-Scaled Units

Data Type: int32

Description: Specifies the units of the values that you want to scale.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Volts</th>
<th>10348</th>
<th>Volts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Amps</td>
<td>10342</td>
<td>Amperes.</td>
</tr>
<tr>
<td>DAQmx_Val_DegF</td>
<td>10144</td>
<td>Degrees Fahrenheit.</td>
</tr>
<tr>
<td>DAQmx_Val_DegC</td>
<td>10143</td>
<td>Degrees Celsius.</td>
</tr>
<tr>
<td>DAQmx_Val_DegR</td>
<td>10145</td>
<td>Degrees Rankine.</td>
</tr>
<tr>
<td>DAQmx_Val_Kelvins</td>
<td>10325</td>
<td>Kelvins.</td>
</tr>
<tr>
<td>DAQmx_Val_Strain</td>
<td>10299</td>
<td>Strain.</td>
</tr>
<tr>
<td>DAQmx_Val_Ohms</td>
<td>10384</td>
<td>Ohms.</td>
</tr>
<tr>
<td>DAQmx_Val_Hz</td>
<td>10373</td>
<td>Hertz.</td>
</tr>
<tr>
<td>DAQmx_Val_Seconds</td>
<td>10364</td>
<td>Seconds.</td>
</tr>
<tr>
<td>DAQmx_Val_Meters</td>
<td>10219</td>
<td>Meters.</td>
</tr>
<tr>
<td>DAQmx_Val_Inches</td>
<td>10379</td>
<td>Inches.</td>
</tr>
<tr>
<td>DAQmx_Val_Degrees</td>
<td>10146</td>
<td>Degrees.</td>
</tr>
<tr>
<td>DAQmx_Val_Radians</td>
<td>10273</td>
<td>Radians.</td>
</tr>
<tr>
<td>DAQmx_Val_g</td>
<td>10186</td>
<td>1 g is approximately equal to 9.81 m/s/s.</td>
</tr>
<tr>
<td>DAQmx_Val_MetersPerSecondSquared</td>
<td>12470</td>
<td>Meters per second per second.</td>
</tr>
<tr>
<td>DAQmx_Val_Pascals</td>
<td>10081</td>
<td>Pascals.</td>
</tr>
<tr>
<td>DAQmx_Val_FromTEDS</td>
<td>12516</td>
<td>Units defined by TEDS information associated with the channel.</td>
</tr>
</tbody>
</table>

You can get/set this property using:

DAQmxGetScalePreScaledUnits
DAQmxSetScalePreScaledUnits
**Scale Type**

**Data Type:** int32

**Description:** Indicates the method or equation form that the custom scale uses.

**Restrictions:** Not Settable

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Linear</td>
<td>10447</td>
<td>Scale values by using the equation $y=mx+b$, where $x$ is a prescaled value and $y$ is a scaled value.</td>
</tr>
<tr>
<td>DAQmx_Val_MapRanges</td>
<td>10448</td>
<td>Scale values proportionally from a range of pre-scaled values to a range of scaled values.</td>
</tr>
<tr>
<td>DAQmx_Val_Polynomial</td>
<td>10449</td>
<td>Scale values by using an Nth order polynomial equation.</td>
</tr>
<tr>
<td>DAQmx_Val_Table</td>
<td>10450</td>
<td>Map an array of pre-scaled values to an array of corresponding scaled values, with all other values scaled proportionally.</td>
</tr>
</tbody>
</table>

You can get this property using:

[DAQmxGetScaleType](#)
**Linear >> Slope**

**Data Type:** float64  

**Description:** Specifies the slope, m, in the equation y=mx+b.

You can get/set this property using:

- `DAQmxGetScaleLinSlope`
- `DAQmxSetScaleLinSlope`
Linear >> Y-Intercept

**Data Type:** float64

**Description:** Specifies the y-intercept, b, in the equation y=mx+b.

You can get/set this property using:

- `DAQmxGetScaleLinYIntercept`
- `DAQmxSetScaleLinYIntercept`
Map >> Scaled Maximum Value

Data Type: float64

Description: Specifies the largest value in the range of scaled values. NI-DAQmx maps this value to Pre-Scaled Maximum Value. Reads coerce samples that are larger than this value to match this value. Writes generate errors for samples that are larger than this value.

You can get/set this property using:

- DAQmxGetScaleMapScaledMax
- DAQmxSetScaleMapScaledMax
Map >> Pre-Scaled Maximum Value

**Data Type:** float64

**Description:** Specifies the largest value in the range of pre-scaled values. NI-DAQmx maps this value to *Scaled Maximum Value*.

You can get/set this property using:

- `DAQmxGetScaleMapPreScaledMax`
- `DAQmxSetScaleMapPreScaledMax`
Map >> Scaled Minimum Value

**Data Type:** float64

**Description:** Specifies the smallest value in the range of scaled values. NI-DAQmx maps this value to Pre-Scaled Minimum Value. Reads coerce samples that are smaller than this value to match this value. Writes generate errors for samples that are smaller than this value.

You can get/set this property using:

- `DAQmxGetScaleMapScaledMin`
- `DAQmxSetScaleMapScaledMin`
Map >> Pre-Scaled Minimum Value

**Data Type:** float64

**Description:** Specifies the smallest value in the range of pre-scaled values. NI-DAQmx maps this value to `Scaled Minimum Value`.

You can get/set this property using:

- `DAQmxGetScaleMapPreScaledMin`
- `DAQmxSetScaleMapPreScaledMin`
Polynomial >> Forward Coefficients

**Data Type:** float64*

**Description:** Specifies an array of coefficients for the polynomial that converts pre-scaled values to scaled values. Each element of the array corresponds to a term of the equation. For example, if index three of the array is 9, the fourth term of the equation is $9x^3$.

You can get/set this property using:

- DAQmxGetScalePolyForwardCoeff
- DAQmxSetScalePolyForwardCoeff
Polynomial >> Reverse Coefficients

**Data Type:** float64*

**Description:** Specifies an array of coefficients for the polynomial that converts scaled values to pre-scaled values. Each element of the array corresponds to a term of the equation. For example, if index three of the array is 9, the fourth term of the equation is 9y^3.

You can get/set this property using:

- `DAQmxGetScalePolyReverseCoeff`
- `DAQmxSetScalePolyReverseCoeff`
Table >> Scaled Values

**Data Type:**  float64*

**Description:** Specifies an array of scaled values. These values map directly to the values in Pre-Scaled Values.

You can get/set this property using:

- DAQmxGetScaleTableScaledVals
- DAQmxSetScaleTableScaledVals
Table >> Pre-Scaled Values

**Data Type:** float64*

**Description:** Specifies an array of pre-scaled values. These values map directly to the values in `Scaled Values`.

You can get/set this property using:

DAQmxGetScaleTablePreScaledVals

DAQmxSetScaleTablePreScaledVals
### Usage

**Data Type:** int32  
**Description:** Specifies how you can use the channel. Using this property acts as a safety mechanism to prevent you from connecting two source channels, for example.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Source</th>
<th>10439</th>
<th>You can use the channel only as an input for a signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Load</td>
<td>10440</td>
<td>You can use the channel only as the output for a signal passing through the switch.</td>
</tr>
<tr>
<td>DAQmx_Val_ReservedForRouting</td>
<td>10441</td>
<td>You can use the channel only to complete routes within a switch.</td>
</tr>
</tbody>
</table>

You can get/set this property using:

- `DAQmxGetSwitchChanUsage`
- `DAQmxSetSwitchChanUsage`
**Capability >> Max AC Carry Current**

**Data Type:** float64

**Description:** Indicates in amperes the maximum AC current that the device can carry.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchChanMaxACCarryCurrent](https://example.com/DAQmxGetSwitchChanMaxACCarryCurrent)
**Capability >> Max AC Switching Current**

**Data Type:** float64

**Description:** Indicates in amperes the maximum AC current that the device can switch. This current is always against an RMS voltage level.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchChanMaxACSwitchCurrent`
Capability >> Max AC Carry Power

**Data Type:** float64

**Description:** Indicates in watts the maximum AC power that the device can carry.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchChanMaxACCarryPwr`
**Capability >> Max AC Switching Power**

**Data Type:** float64

**Description:** Indicates in watts the maximum AC power that the device can switch.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSwitchChanMaxACSwitchPwr
Capability >> Max DC Carry Current

**Data Type:** float64

**Description:** Indicates in amperes the maximum DC current that the device can carry.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchChanMaxDCCarryCurrent`
Capability >> Max DC Switching Current

**Data Type:** float64

**Description:** Indicates in amperes the maximum DC current that the device can switch. This current is always against a DC voltage level.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSwitchChanMaxDCSwitchCurrent
**Capability >> Max DC Carry Power**

**Data Type:** float64

**Description:** Indicates in watts the maximum DC power that the device can carry.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchChanMaxDCarryPwr`
Capability >> Max DC Switching Power

**Data Type:** float64  
**Description:** Indicates in watts the maximum DC power that the device can switch.  
**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchChanMaxDCSwitchPwr](#)
**Capability >> Max AC Voltage**

**Data Type:** float64

**Description:** Indicates in volts the maximum AC RMS voltage that the device can switch.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchChanMaxACVoltage`
**Capability >> Max DC Voltage**

**Data Type:** float64

**Description:** Indicates in volts the maximum DC voltage that the device can switch.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchChanMaxDCVoltage](#)
Capability >> Wire Mode

**Data Type:**  uInt32

**Description:**  Indicates the number of wires that the channel switches.

**Restrictions:**  Not Settable

You can get this property using:

DAQmxGetSwitchChanWireMode
**Capability >> Bandwidth**

**Data Type:** float64

**Description:** Indicates in Hertz the maximum frequency of a signal that can pass through the switch without significant deterioration.

**Restrictions:** Not Settable

You can get this property using:

Capability >> Impedance

**Data Type**: float64

**Description**: Indicates in ohms the switch impedance. This value is important in the RF domain and should match the impedance of the sources and loads.

**Restrictions**: Not Settable

You can get this property using:

DAQmxGetSwitchChanImpedance
**Settling Time**

**Data Type:** float64

**Description:** Specifies in seconds the amount of time to wait for the switch to settle (or debounce). NI-DAQmx adds this time to the settling time of the motherboard. Modify this property only if the switch does not settle within the settling time of the motherboard. Refer to device documentation for supported settling times.

You can get/set this property using:

- `DAQmxGetSwitchDevSettlingTime`
- `DAQmxSetSwitchDevSettlingTime`
Auto Connect Analog Bus

**Data Type:** bool32

**Description:** Specifies if NI-DAQmx routes multiplexed channels to the analog bus backplane. Only the SCXI-1127 and SCXI-1128 support this property.

You can get/set this property using:

- `DAQmxGetSwitchDevAutoConnAnlgBus`
- `DAQmxSetSwitchDevAutoConnAnlgBus`
Power Down Latching Relays After Settling

Data Type: bool32

Description: Specifies if DAQmxSwitchWaitForSettling() powers down latching relays after waiting for the device to settle.

You can get/set this property using:

DAQmxGetSwitchDevPwrDownLatchRelaysAfterSettling
DAQmxSetSwitchDevPwrDownLatchRelaysAfterSettling
Is Settled

**Data Type:** bool32

**Description:** Indicates when *Settling Time* expires.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchDevSettled`
**Capability >> Relay List**

**Data Type:** char*

**Description:** Indicates a comma-delimited list of relay names.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchDevRelayList](#)
Capability >> Number of Relays

**Data Type:**  uInt32

**Description:** Indicates the number of relays on the device. This value matches the number of relay names in Relay List.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchDevNumRelays](https://example.com/DAQmxGetSwitchDevNumRelays)
Capability >> Switch Channel List

**Data Type:** char*

**Description:** Indicates a comma-delimited list of channel names for the current topology of the device.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchDevSwitchChanList`
**Capability >> Number of Switch Channels**

**Data Type:** uInt32

**Description:** Indicates the number of switch channels for the current topology of the device. This value matches the number of channel names in Switch Channel List.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchDevNumSwitchChans](#)
**Capability >> Number of Rows**

**Data Type:**  uInt32

**Description:** Indicates the number of rows on a device in a matrix switch topology. Indicates the number of multiplexed channels on a device in a mux topology.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSwitchDevNumRows
Capability >> Number of Columns

**Data Type:**  ulInt32

**Description:** Indicates the number of columns on a device in a matrix switch topology. This value is always 1 if the device is in a mux topology.

**Restrictions:**  Not Settable

You can get this property using:

DAQmxGetSwitchDevNumColumns
Topology

**Data Type:** char*

**Description:** Indicates the current topology of the device. This value is one of the topology options in DAQmxSwitchSetTopologyAndReset().

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSwitchDevTopology`
Break Mode

**Data Type:** int32

**Description:** Specifies the action to take between each entry in a scan list.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_NoAction</th>
<th>10227</th>
<th>When advancing to the next entry in the scan list, leave all previous connections intact.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_BreakBeforeMake</td>
<td>10110</td>
<td>When advancing to the next entry in the scan list, disconnect all previous connections before making any new connections.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetSwitchScanBreakMode`
- `DAQmxSetSwitchScanBreakMode`
- `DAQmxResetSwitchScanBreakMode`
Repeat Mode

**Data Type:** int32

**Description:** Specifies if the task advances through the scan list multiple times.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Finite</td>
<td>10172</td>
<td>The task advances through the scan list one time only. NI-DAQmx ignores any Advance Triggers after completing the scan list.</td>
</tr>
<tr>
<td>DAQmx_Val_Cont</td>
<td>10117</td>
<td>The task returns to the beginning of the scan list when it reaches the end of the scan list.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetSwitchScanRepeatMode`
- `DAQmxSetSwitchScanRepeatMode`
- `DAQmxResetSwitchScanRepeatMode`
Is Waiting For Advance

**Data Type:** bool32

**Description:** Indicates if the switch hardware is waiting for an Advance Trigger. If the hardware is waiting, it completed the previous entry in the scan list.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSwitchScanWaitingForAdv](https://www.ni.com/driverAXB.html?lang=en&product=mxgetswitchscanwaitingforadv)
Global Channels

**Data Type:** char*

**Description:** Indicates an array that contains the names of all global channels saved on the system.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetSysGlobalChans](https://www.ni.com/)

Scales

**Data Type:** char*

**Description:** Indicates an array that contains the names of all custom scales saved on the system.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSysScales
Tasks

**Data Type:** char*

**Description:** Indicates an array that contains the names of all tasks saved on the system.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSysTasks
Device Names

**Data Type:**  char*

**Description:** Indicates the names of all devices installed in the system.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSysDevNames
System >> NI-DAQ Version >> Major

Data Type: uInt32
Description: Indicates the major portion of the installed version of NI-DAQ, such as 7 for version 7.0.
Restrictions: Not Settable

You can get this property using:

DAQmxGetSysNIDAQMajorVersion
System >> NI-DAQ Version >> Minor

**Data Type:** uInt32

**Description:** Indicates the minor portion of the installed version of NI-DAQ, such as 0 for version 7.0.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSysNIDAQMinorVersion`
Name

**Data Type:** char*

**Description:** Indicates the name of the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetTaskName](#)
Channels

**Data Type:** char*

**Description:** Indicates the names of all virtual channels in the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetTaskChannels](#)
Number of Channels

**Data Type:** uInt32

**Description:** Indicates the number of virtual channels in the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetTaskNumChans](#)
Devices

**Data Type:** char*

**Description:** Indicates an array containing the names of all devices in the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetTaskDevices](#)
**Number of Devices**

**Data Type:** uInt32  
**Description:** Indicates the number of devices in the task.  
**Restrictions:** Not Settable

You can get this property using:  

[DAQmxGetTaskNumDevices](http://example.com/DAQmxGetTaskNumDevices)
Task Done

**Data Type:** bool32

**Description:** Indicates whether the task [completed execution](https://example.com).

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetTaskComplete](https://example.com)
Sample Quantity >> Sample Mode

**Data Type:** int32

**Description:** Specifies if a task acquires or generates a finite number of samples or if it continuously acquires or generates samples.

For an analog input task with a Reference Trigger, you must set this property to DAQmx_Val_FiniteSamps even though the task runs continuously until the Reference Trigger occurs.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_FiniteSamps</td>
<td>10178</td>
<td>Acquire or generate a finite number of samples.</td>
</tr>
<tr>
<td>DAQmx_Val_ContSamps</td>
<td>10123</td>
<td>Acquire or generate samples until you stop the task.</td>
</tr>
<tr>
<td>DAQmx_Val_HWTimedSinglePoint</td>
<td>12522</td>
<td>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.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetSampQuantSampMode
- DAQmxSetSampQuantSampMode
- DAQmxResetSampQuantSampMode
Sample Quantity >> Samples Per Channel

**Data Type:** uInt64

**Description:** Specifies the number of samples to acquire or generate for each channel if Sample Mode is DAQmx_Val_FiniteSamps. If Sample Mode is DAQmx_Val_ContSamps, NI-DAQmx uses this value to determine the buffer size.

To configure an analog output task to generate a finite number of cycles of a waveform, set this property to (desired number of cycles) * (number of samples per cycle).

This property affects the buffer allocation for the task.

You can get/set/reset this property using:

- DAQmxGetSampQuantSampPerChan
- DAQmxSetSampQuantSampPerChan
- DAQmxResetSampQuantSampPerChan
Sample Timing Type

**Data Type:** int32

**Description:** Specifies the type of sample timing to use for the task.

- Select DAQmx_Val_SampClk when a hardware signal (usually a clock) must acquire or produce samples. To perform buffered edge counting, for example, select DAQmx_Val_SampClk and use `Source` to specify the source of the Sample Clock.

- Select DAQmx_Val_Handshake when you want to use bidirectional hardware signals to time the exchange of digital data between two devices.

- Select DAQmx_Val_OnDemand when you want to acquire data only when an NI-DAQmx Read function executes or to generate data only when an NI-DAQmx Write function executes.

- Select DAQmx_Val_Implicit when you perform a buffered period or frequency counter measurement or when you generate a finite pulse train.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_SampClk</th>
<th>10388</th>
<th>Use a sample clock to determine how often to acquire or generate samples.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_BurstHandshake</td>
<td>12548</td>
<td>Determine sample timing using burst handshaking between the device and a peripheral device.</td>
</tr>
<tr>
<td>DAQmx_Val_Handshake</td>
<td>10389</td>
<td>Determine sample timing by using digital handshaking between the device and a peripheral device.</td>
</tr>
<tr>
<td>DAQmx_Val_Implicit</td>
<td>10451</td>
<td>Configure only the duration of the task.</td>
</tr>
<tr>
<td>DAQmx_Val_OnDemand</td>
<td>10390</td>
<td>Acquire or generate a sample on each read or write operation.</td>
</tr>
<tr>
<td>DAQmx_Val_ChangeDetection</td>
<td>12504</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

You can get/set/reset this property using:

- DAQmxGetSampTimingType
- DAQmxSetSampTimingType
- DAQmxResetSampTimingType
Sample Clock >> Rate

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetSampClkRate`
- `DAQmxSetSampClkRate`
- `DAQmxResetSampClkRate`
Sample Clock >> Maximum Rate

**Data Type:** float64

**Description:** 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.

For multiplexed devices, NI-DAQmx calculates the maximum sample clock rate based on the maximum AI Convert Clock rate unless you set Rate. If you set that property, NI-DAQmx calculates the maximum sample clock rate based on that setting. Use Maximum Rate to query the maximum AI Convert Clock rate. NI-DAQmx also uses the minimum sample clock delay to calculate the maximum sample clock rate unless you set Delay.

For simultaneous sampling devices, the maximum Sample Clock rate is the maximum rate of the ADC.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetSampClikMaxRate
**Sample Clock >> Source**

**Data Type:** char*

**Description:** Specifies the terminal of the signal to use as the Sample Clock.

You can get/set/reset this property using:

- `DAQmxGetSampClkSrc`
- `DAQmxSetSampClkSrc`
- `DAQmxResetSampClkSrc`
**Sample Clock >> Active Edge**

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Decimal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetSampClkActiveEdge
- DAQmxSetSampClkActiveEdge
- DAQmxResetSampClkActiveEdge
Sample Clock >> Underflow Behavior

**Data Type:** int32  
**Description:** Specifies the action to take when the onboard memory of the device becomes empty.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_HaltOutputAndError</td>
<td>14615</td>
<td>Stop generating samples and return an error.</td>
</tr>
<tr>
<td>DAQmx_Val_PauseUntilDataAvailable</td>
<td>14616</td>
<td>Pause the task until samples are available in the FIFO.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetSampClkUnderflowBehavior`
- `DAQmxSetSampClkUnderflowBehavior`
- `DAQmxResetSampClkUnderflowBehavior`
Sample Clock >> Timebase Divisor

**Data Type:** uInt32

**Description:** Specifies the number of Sample Clock Timebase pulses needed to produce a single Sample Clock pulse.

The rate of the Sample Clock is equal to (frequency of Sample Clock Timebase) / (value of this property).

If the Sample Clock Timebase is not a periodic clock, the value of this property determines the number of Sample Clock Timebase edges that the device must receive before producing each Sample Clock pulse. Use **Active Edge** to specify the polarity of these edges.

Setting this property has a similar effect to setting **Rate**. Use **Rate** when you know the rate of the Sample Clock Timebase and you want to acquire or generate samples at the specified rate. Use this property when you have an external timebase that you want to divide down and use as the Sample Clock, but you do not know rate of the external timebase.

You can get/set/reset this property using:

- DAQmxGetSampClkTimebaseDiv
- DAQmxSetSampClkTimebaseDiv
- DAQmxResetSampClkTimebaseDiv
Sample Clock >> Timebase >> Rate

Data Type: float64

Description: 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.

You can get/set/reset this property using:

DAQmxGetSampClkTimebaseRate
DAQmxSetSampClkTimebaseRate
DAQmxResetSampClkTimebaseRate
Sample Clock >> Timebase >> Source

Data Type: char*
Description: Specifies the terminal of the signal to use as the Sample Clock Timebase.

You can get/set/reset this property using:

DAQmxGetSampClkTimebaseSrc
DAQmxSetSampClkTimebaseSrc
DAQmxResetSampClkTimebaseSrc
Sample Clock >> Timebase >> Active Edge

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetSampClkTimebaseActiveEdge
- DAQmxSetSampClkTimebaseActiveEdge
- DAQmxResetSampClkTimebaseActiveEdge
**Sample Clock >> Timebase >> Master Timebase Divisor**

**Data Type:** uInt32

**Description:** Specifies the number of pulses of the Master Timebase needed to produce a single pulse of the Sample Clock Timebase.

The rate of the Sample Clock Timebase is equal to (frequency of Master Timebase) / (value of this property). The only valid values for this property are 1 and 200.

Setting this property has a similar effect to setting Rate. Use Rate when you know the rate of the Master Timebase and you want to produce a Sample Clock Timebase at the specified rate. Use this property when you have an external Master Timebase that you want to divide down and use as the Sample Clock Timebase, but you do not know rate of that external Master Timebase.

You can get/set/reset this property using:

- `DAQmxGetSampClkTimebaseMasterTimebaseDiv`
- `DAQmxSetSampClkTimebaseMasterTimebaseDiv`
- `DAQmxResetSampClkTimebaseMasterTimebaseDiv`
Sample Clock >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- DAQmxGetSampClkDigFltrEnable
- DAQmxSetSampClkDigFltrEnable
- DAQmxResetSampClkDigFltrEnable
Sample Clock >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetSampClkDigFltrMinPulseWidth`
- `DAQmxSetSampClkDigFltrMinPulseWidth`
- `DAQmxResetSampClkDigFltrMinPulseWidth`
**Sample Clock >> Digital Filter >> Timebase >> Source**

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- DAQmxGetSampClkDigFiltrTimebaseSrc
- DAQmxSetSampClkDigFiltrTimebaseSrc
- DAQmxResetSampClkDigFiltrTimebaseSrc
Sample Clock >> Digital Filter >> Timebase >> Rate

**Data Type:**  float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetSampClkDigFltrTimebaseRate`
- `DAQmxSetSampClkDigFltrTimebaseRate`
- `DAQmxResetSampClkDigFltrTimebaseRate`
**Sample Clock >> Digital Synchronization >> Enable**

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetSampClkDigSyncEnable`
- `DAQmxSetSampClkDigSyncEnable`
- `DAQmxResetSampClkDigSyncEnable`
Handshake >> Delay After Transfer

**Data Type:** float64

**Description:** Specifies the number of seconds to wait after a handshake cycle before starting a new handshake cycle.

You can get/set/reset this property using:

- `DAQmxGetHshkDelayAfterXfer`
- `DAQmxSetHshkDelayAfterXfer`
- `DAQmxResetHshkDelayAfterXfer`
**Handshake >> Start Condition**

**Data Type:** int32  
**Description:** Specifies the point in the handshake cycle that the device is in when the task starts.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Immediate</th>
<th>10198</th>
<th>Device is waiting for space in the FIFO (for acquisition) or waiting for samples (for generation).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_WaitForHandshakeTriggerAssert</td>
<td>12550</td>
<td>Device is waiting for the Handshake Trigger to assert.</td>
</tr>
<tr>
<td>DAQmx_Val_WaitForHandshakeTriggerDeassert</td>
<td>12551</td>
<td>Device is waiting for the Handshake Trigger to deassert.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetHshkStartCond
- DAQmxSetHshkStartCond
- DAQmxResetHshkStartCond
Handshake >> Sample Input Data When

Data Type: int32

Description: Specifies on which edge of the Handshake Trigger an input task latches the data from the peripheral device.

Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_HandshakeTriggerAsserts 12552</td>
<td>Latch data when the Handshake Trigger asserts.</td>
</tr>
<tr>
<td>DAQmx_Val_HandshakeTriggerDeasserts 12553</td>
<td>Latch data when the Handshake Trigger deasserts.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetHshkSampleInputDataWhen
- DAQmxSetHshkSampleInputDataWhen
- DAQmxResetHshkSampleInputDataWhen
Change Detection >> Digital Input >> Rising Edge Physical Channels

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetChangeDetectDIRisingEdgePhysicalChans
- DAQmxSetChangeDetectDIRisingEdgePhysicalChans
- DAQmxResetChangeDetectDIRisingEdgePhysicalChans
### Change Detection >> Digital Input >> Falling Edge Physical Channels

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetChangeDetectDIFallingEdgePhysicalChans`
- `DAQmxSetChangeDetectDIFallingEdgePhysicalChans`
- `DAQmxResetChangeDetectDIFallingEdgePhysicalChans`
On Demand >> Simultaneous Analog Output Enable

**Data Type:** bool32

**Description:** Specifies whether to update all channels in the task simultaneously, rather than updating channels independently when you write a sample to that channel.

You can get/set/reset this property using:

- `DAQmxGetOnDemandSimultaneousAOEnable`
- `DAQmxSetOnDemandSimultaneousAOEnable`
- `DAQmxResetOnDemandSimultaneousAOEnable`
**More >> AI Convert >> Rate**

**Data Type:** float64

**Description:** 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.

By default, NI-DAQmx selects the maximum convert rate supported by the device, plus up to 10 microseconds per channel settling time. Other task settings, such as high channel counts or setting Delay, can result in a faster default convert rate.

CompactDAQ applies up to 10 microseconds of settling time for all C Series devices even if the maximum convert rates of the devices differ.

If you connect signal conditioning accessories with track and hold capabilities, such as an SCXI module, to the device, NI-DAQmx uses the fastest convert rate possible that meets the settling requirements for the slowest module sampled. Refer to the device documentation for the signal conditioning accessory for more information.

You can get/set/reset this property using:

- DAQmxGetAIConvRate
- DAQmxSetAIConvRate
- DAQmxResetAIConvRate
- DAQmxGetAIConvRateEx
- DAQmxSetAIConvRateEx
- DAQmxResetAIConvRateEx
More >> AI Convert >> Maximum Rate

**Data Type:**  float64

**Description:** Indicates the maximum convert rate supported by the task, given the current devices and channel count.

This rate is generally faster than the default AI Convert Clock rate selected by NI-DAQmx, because NI-DAQmx adds in an additional 10 microseconds per channel settling time to compensate for most potential system settling constraints.

For single channel tasks, the maximum AI Convert Clock rate is the maximum rate of the ADC. For multiple channel tasks, the maximum AI Convert Clock rate is the maximum convert rate of the analog hardware, including the ADC, filters, multiplexers, and amplifiers. Signal conditioning accessories can further constrain the maximum AI Convert Clock based on timing and settling requirements.

**Restrictions:** Not Settable

You can get this property using:

- `DAQmxGetAIConvMaxRate`
- `DAQmxGetAIConvMaxRateEx`
More >> AI Convert >> Source

**Data Type:** char*

**Description:** Specifies the terminal of the signal to use as the AI Convert Clock.

You can get/set/reset this property using:

- `DAQmxGetAIConvSrc`
- `DAQmxSetAIConvSrc`
- `DAQmxResetAIConvSrc`
- `DAQmxGetAIConvSrcEx`
- `DAQmxSetAIConvSrcEx`
- `DAQmxResetAIConvSrcEx`
More >> AI Convert >> Active Edge

**Data Type:** int32

**Description:** Specifies on which edge of the clock pulse an analog-to-digital conversion takes place.

This property is useful primarily when the signal you use as the AI Convert Clock is not a periodic clock. For example, set this property to DAQmx_Val_Rising to perform an analog to digital conversion on each rising edge of an aperiodic signal.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIConvActiveEdge
- DAQmxSetAIConvActiveEdge
- DAQmxResetAIConvActiveEdge
- DAQmxGetAIConvActiveEdgeEx
- DAQmxSetAIConvActiveEdgeEx
- DAQmxResetAIConvActiveEdgeEx
More >> AI Convert >> Timebase Divisor

**Data Type:** uInt32

**Description:** Specifies the number of AI Convert Clock Timebase pulses needed to produce a single AI Convert Clock pulse.

You can get/set/reset this property using:

- DAQmxGetAIConvTimebaseDiv
- DAQmxSetAIConvTimebaseDiv
- DAQmxResetAIConvTimebaseDiv
- DAQmxGetAIConvTimebaseDivEx
- DAQmxSetAIConvTimebaseDivEx
- DAQmxResetAIConvTimebaseDivEx
More >> AI Convert >> Timebase >> Source

**Data Type:** int32

**Description:** Specifies the terminal of the signal to use as the AI Convert Clock Timebase.

### Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_SameAsSampTimebase</td>
<td>Use the same source as Sample Clock timebase.</td>
</tr>
<tr>
<td>DAQmx_Val_SameAsMasterTimebase</td>
<td>Use the same source as the Master Timebase.</td>
</tr>
<tr>
<td>DAQmx_Val_20MHzTimebase</td>
<td>Use the onboard 20MHz timebase.</td>
</tr>
<tr>
<td>DAQmx_Val_80MHzTimebase</td>
<td>Use the onboard 80MHz timebase.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAIConvTimebaseSrc
- DAQmxSetAIConvTimebaseSrc
- DAQmxResetAIConvTimebaseSrc
- DAQmxGetAIConvTimebaseSrcEx
- DAQmxSetAIConvTimebaseSrcEx
- DAQmxResetAIConvTimebaseSrcEx
More >> AI Convert >> Delay From Sample Clock >>

Delay Units

**Data Type:** int32

**Description:** Specifies the units of Delay.

NI-DAQmx uses the AI Convert Clock timebase to produce the delay.

If you set this property to DAQmx_Val_Ticks and N is the value of Delay, N pulses of the AI Convert Clock Timebase occur after receiving a Sample Clock pulse before the acquisition of the sample begins.

If you set this property to DAQmx_Val_Seconds, that number of seconds elapses after receiving a Sample Clock pulse before the acquisition of the sample begins.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_Seconds</th>
<th>10364</th>
<th>Seconds.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDelayFromSampClkDelayUnits
- DAQmxSetDelayFromSampClkDelayUnits
- DAQmxResetDelayFromSampClkDelayUnits
- DAQmxGetDelayFromSampClkDelayUnitsEx
- DAQmxSetDelayFromSampClkDelayUnitsEx
- DAQmxResetDelayFromSampClkDelayUnitsEx
More >> AI Convert >> Delay From Sample Clock >> Delay

**Data Type:** float64

**Description:** 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 Units](#).

You can get/set/reset this property using:

- `DAQmxGetDelayFromSampClkDelay`
- `DAQmxSetDelayFromSampClkDelay`
- `DAQmxResetDelayFromSampClkDelay`
- `DAQmxGetDelayFromSampClkDelayEx`
- `DAQmxSetDelayFromSampClkDelayEx`
- `DAQmxResetDelayFromSampClkDelayEx`
More >> Master Timebase >> Rate

**Data Type:** float64

**Description:** Specifies the rate of the Master Timebase.

This property is useful only when the source of the Master Timebase is not the onboard 20 MHz timebase. NI-DAQmx requires the Master Timebase rate to calculate other timing parameters.

You can get/set/reset this property using:

- DAQmxGetMasterTimebaseRate
- DAQmxSetMasterTimebaseRate
- DAQmxResetMasterTimebaseRate
More >> Master Timebase >> Source

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetMasterTimebaseSrc`
- `DAQmxSetMasterTimebaseSrc`
- `DAQmxResetMasterTimebaseSrc`
More >> Reference Clock >> Rate

**Data Type**: float64

**Description**: Specifies the frequency of the Reference Clock.

You can get/set/reset this property using:

- `DAQmxGetRefClkRate`
- `DAQmxSetRefClkRate`
- `DAQmxResetRefClkRate`
More >> Reference Clock >> Source

**Data Type:** char*

**Description:** Specifies the *terminal* of the signal to use as the Reference Clock.

You can get/set/reset this property using:

- `DAQmxGetRefClkSrc`
- `DAQmxSetRefClkSrc`
- `DAQmxResetRefClkSrc`
More >> Synchronization Pulse >> Source

**Data Type:**  char*

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetSyncPulseSrc`
- `DAQmxSetSyncPulseSrc`
- `DAQmxResetSyncPulseSrc`
More >> Synchronization Pulse >> Synchronization Time

**Data Type:** float64

**Description:** Indicates in seconds the delay required to reset the ADCs/DACs after the device receives the synchronization pulse. This time is the minimum delay required by the device between the receipt of the synchronization pulse and the start of the acquisition. Read this property for all slave devices and set Minimum Delay To Start for the master device to the maximum of these delays.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetSyncPulseSyncTime`
More >> Synchronization Pulse >> Minimum Delay To Start

**Data Type:** float64

**Description:** Specifies in seconds the amount of time that elapses after the master device issues the synchronization pulse before the task starts.

Read [Synchronization Time](#) for all slave devices, and set this property for the master device to the maximum of those values.

You can get/set/reset this property using:

- `DAQmxGetSyncPulseMinDelayToStart`
- `DAQmxSetSyncPulseMinDelayToStart`
- `DAQmxResetSyncPulseMinDelayToStart`
**Advanced >> Sample Timing Engine**

**Data Type:** uInt32

**Description:** Specifies which timing engine to use for the specified timing type. Refer to device documentation for information on supported timing engines.

You can get/set/reset this property using:

- DAQmxGetSampTimingEngine
- DAQmxSetSampTimingEngine
- DAQmxResetSampTimingEngine
**Start >> Trigger Type**

**Data Type:** int32

**Description:** Specifies the type of trigger to use to start a task.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AnlgEdge</td>
<td>10099</td>
<td>Trigger when an analog signal signal crosses a threshold.</td>
</tr>
<tr>
<td>DAQmx_Val_DigEdge</td>
<td>10150</td>
<td>Trigger on the rising or falling edge of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_DigPattern</td>
<td>10398</td>
<td>Trigger when digital physical channels match a digital pattern.</td>
</tr>
<tr>
<td>DAQmx_Val_AnlgWin</td>
<td>10103</td>
<td>Trigger when an analog signal enters or leaves a range of values. The range is in the units of the measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Disable triggering for the task.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetStartTrigType`
- `DAQmxSetStartTrigType`
- `DAQmxResetStartTrigType`
Start >> Digital Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a terminal where there is a digital signal to use as the source of the Start Trigger.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeStartTrigSrc`
- `DAQmxSetDigEdgeStartTrigSrc`
- `DAQmxResetDigEdgeStartTrigSrc`
Start >> Digital Edge >> Edge

Data Type: int32

Description: Specifies on which edge of a digital pulse to start acquiring or generating samples.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Rising</th>
<th>10280</th>
<th>Rising edge(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDigEdgeStartTrigEdge
- DAQmxSetDigEdgeStartTrigEdge
- DAQmxResetDigEdgeStartTrigEdge
Start >> Digital Edge >> Digital Filter >> Enable

**Data Type:**  bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- DAQmxGetDigEdgeStartTrigDigFtrEnable
- DAQmxSetDigEdgeStartTrigDigFtrEnable
- DAQmxResetDigEdgeStartTrigDigFtrEnable
Start >> Digital Edge >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64  
**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeStartTrigDigFltrMinPulseWidth`
- `DAQmxSetDigEdgeStartTrigDigFltrMinPulseWidth`
- `DAQmxResetDigEdgeStartTrigDigFltrMinPulseWidth`
Start >> Digital Edge >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeStartTrigDigFltrTimebaseSrc`
- `DAQmxSetDigEdgeStartTrigDigFltrTimebaseSrc`
- `DAQmxResetDigEdgeStartTrigDigFltrTimebaseSrc`
Start >> Digital Edge >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

`DAQmxGetDigEdgeStartTrigDigFltrTimebaseRate`

`DAQmxSetDigEdgeStartTrigDigFltrTimebaseRate`

`DAQmxResetDigEdgeStartTrigDigFltrTimebaseRate`
Start >> Digital Edge >> Digital Synchronization >>
Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetDigEdgeStartTrigDigSyncEnable
- DAQmxSetDigEdgeStartTrigDigSyncEnable
- DAQmxResetDigEdgeStartTrigDigSyncEnable
Start >> Digital Pattern >> Source

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetDigPatternStartTrigSrc`
- `DAQmxSetDigPatternStartTrigSrc`
- `DAQmxResetDigPatternStartTrigSrc`
Start >> Digital Pattern >> Pattern

**Data Type:** char*

**Description:** Specifies the digital pattern that must be met for the Start Trigger to occur.

You can get/set/reset this property using:

`DAQmxGetDigPatternStartTrigPattern`
`DAQmxSetDigPatternStartTrigPattern`
`DAQmxResetDigPatternStartTrigPattern`
**Start >> Digital Pattern >> Trigger When**

**Data Type:** int32

**Description:** Specifies whether the Start Trigger occurs when the physical channels specified with Source match or differ from the digital pattern specified with Pattern.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_PatternMatches</td>
<td>10254</td>
<td>Trigger when the physical channels match the specified pattern.</td>
</tr>
<tr>
<td>DAQmx_Val_PatternDoesNotMatch</td>
<td>10253</td>
<td>Trigger when the physical channels do not match the specified pattern.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDigPatternStartTrigWhen
- DAQmxSetDigPatternStartTrigWhen
- DAQmxResetDigPatternStartTrigWhen
Start >> Analog Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Start Trigger.

For E Series devices, if you use a channel name, the channel must be the first one in the task. The only terminal you can use for E Series devices is PFI0.

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeStartTrigSrc`
- `DAQmxSetAnlgEdgeStartTrigSrc`
- `DAQmxResetAnlgEdgeStartTrigSrc`
Start >> Analog Edge >> Slope

**Data Type:** int32

**Description:** Specifies on which slope of the trigger signal to start acquiring or generating samples.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_RisingSlope</td>
<td>10280</td>
<td>Trigger on the rising slope of the signal.</td>
</tr>
<tr>
<td>DAQmx_Val_FallingSlope</td>
<td>10171</td>
<td>Trigger on the falling slope of the signal.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeStartTrigSlope`
- `DAQmxSetAnlgEdgeStartTrigSlope`
- `DAQmxResetAnlgEdgeStartTrigSlope`
Start >> Analog Edge >> Level

**Data Type:**  float64

**Description:** Specifies at what threshold in the units of the measurement or generation to start acquiring or generating samples. Use `slope` to specify on which slope to trigger on this threshold.

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeStartTrigLvl`
- `DAQmxSetAnlgEdgeStartTrigLvl`
- `DAQmxResetAnlgEdgeStartTrigLvl`
Start >> Analog Edge >> Hysteresis

Data Type: float64

Description: Specifies a hysteresis level in the units of the measurement or generation. If Slope is DAQmx_Val_RisingSlope, the trigger does not deassert until the source signal passes below Level minus the hysteresis. If Slope is DAQmx_Val_FallingSlope, the trigger does not deassert until the source signal passes above Level plus the hysteresis.

You can get/set/reset this property using:

DAQmxGetAnlgEdgeStartTrigHyst
DAQmxGetAnlgEdgeStartTrigHyst
DAQmxResetAnlgEdgeStartTrigHyst
**Start >> Analog Edge >> Coupling**

**Data Type:** int32

**Description:** Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Alternating Current.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Direct Current.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeStartTrigCoupling`
- `DAQmxSetAnlgEdgeStartTrigCoupling`
- `DAQmxResetAnlgEdgeStartTrigCoupling`
Start >> Analog Window >> Source

**Data Type:** char*

**Description:** Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Start Trigger.

For E Series devices, if you use a channel name, the channel must be the first one in the task. The only terminal you can use for E Series devices is PFI0.

You can get/set/reset this property using:

- `DAQmxGetAnlgWinStartTrigSrc`
- `DAQmxSetAnlgWinStartTrigSrc`
- `DAQmxResetAnlgWinStartTrigSrc`
Start >> Analog Window >> Trigger When

**Data Type:** int32

**Description:** Specifies whether the task starts acquiring or generating samples when the signal enters or leaves the window you specify with Bottom and Top.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_EnteringWin</th>
<th>10163</th>
<th>Trigger when the signal enters the window.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_LeavingWin</td>
<td>10208</td>
<td>Trigger when the signal leaves the window.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAnlgWinStartTrigWhen
- DAQmxSetAnlgWinStartTrigWhen
- DAQmxResetAnlgWinStartTrigWhen
Start >> Analog Window >> Top

**Data Type:** float64

**Description:** Specifies the upper limit of the window. Specify this value in the units of the measurement or generation.

You can get/set/reset this property using:

- DAQmxGetAnlgWinStartTrigTop
- DAQmxSetAnlgWinStartTrigTop
- DAQmxResetAnlgWinStartTrigTop
Start >> Analog Window >> Bottom

**Data Type:** float64

**Description:** Specifies the lower limit of the window. Specify this value in the units of the measurement or generation.

You can get/set/reset this property using:

- DAQmxGetAnlgWinStartTrigBtm
- DAQmxSetAnlgWinStartTrigBtm
- DAQmxResetAnlgWinStartTrigBtm
Start >> Analog Window >> Coupling

**Data Type:** int32

**Description:** Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Alternating Current.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Direct Current.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAnlgWinStartTrigCoupling
- DAQmxSetAnlgWinStartTrigCoupling
- DAQmxResetAnlgWinStartTrigCoupling
Start >> More >> Delay

**Data Type:** float64

**Description:** 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](#).

You can get/set/reset this property using:

- `DAQmxGetStartTrigDelay`
- `DAQmxSetStartTrigDelay`
- `DAQmxResetStartTrigDelay`
**Start >> More >> Delay Units**

**Data Type:**  int32

**Description:** Specifies the units of Delay.

**Valid values**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_SampClkPeriods</td>
<td>10286</td>
<td>Complete periods of the Sample Clock.</td>
</tr>
<tr>
<td>DAQmx_Val_Seconds</td>
<td>10364</td>
<td>Seconds.</td>
</tr>
<tr>
<td>DAQmx_Val_Ticks</td>
<td>10304</td>
<td>Timebase ticks.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetStartTrigDelayUnits`
- `DAQmxSetStartTrigDelayUnits`
- `DAQmxResetStartTrigDelayUnits`
**Start >> More >> Retriggerable**

**Data Type:** bool32

**Description:** Specifies whether to enable retriggerable counter pulse generation. When you set this property to TRUE, the device generates pulses each time it receives a trigger. The device ignores a trigger if it is in the process of generating pulses.

You can get/set/reset this property using:

- `DAQmxGetStartTrigRetriggerable`
- `DAQmxSetStartTrigRetriggerable`
- `DAQmxResetStartTrigRetriggerable`
Reference >> Trigger Type

Data Type: int32
Description: Specifies the type of trigger to use to mark a reference point for the measurement.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_AnlgEdge</th>
<th>10099</th>
<th>Trigger when an analog signal signal crosses a threshold.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DigEdge</td>
<td>10150</td>
<td>Trigger on the rising or falling edge of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_DigPattern</td>
<td>10398</td>
<td>Trigger when digital physical channels match a digital pattern.</td>
</tr>
<tr>
<td>DAQmx_Val_AnlgWin</td>
<td>10103</td>
<td>Trigger when an analog signal enters or leaves a range of values. The range is in the units of the measurement.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Disable triggering for the task.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetRefTrigType
- DAQmxSetRefTrigType
- DAQmxResetRefTrigType
Reference >> Pretrigger Samples per Channel

**Data Type:** uInt32

**Description:** 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 [Samples Per Channel](#) minus the number of pretrigger samples per channel.

You can get/set/reset this property using:

- `DAQmxGetRefTrigPretrigSamples`
- `DAQmxSetRefTrigPretrigSamples`
- `DAQmxResetRefTrigPretrigSamples`
Reference >> Digital Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a terminal where there is a digital signal to use as the source of the Reference Trigger.

You can get/set/reset this property using:

- DAQmxGetDigEdgeRefTrigSrc
- DAQmxSetDigEdgeRefTrigSrc
- DAQmxResetDigEdgeRefTrigSrc
Reference >> Digital Edge >> Edge

Data Type: int32

Description: Specifies on what edge of a digital pulse the Reference Trigger occurs.

Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDigEdgeRefTrigEdge
- DAQmxSetDigEdgeRefTrigEdge
- DAQmxResetDigEdgeRefTrigEdge
**Reference >> Digital Pattern >> Source**

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- DAQmxGetDigPatternRefTrigSrc
- DAQmxSetDigPatternRefTrigSrc
- DAQmxResetDigPatternRefTrigSrc
Reference >> Digital Pattern >> Pattern

**Data Type:** char*

**Description:** Specifies the digital pattern that must be met for the Reference Trigger to occur.

You can get/set/reset this property using:

- `DAQmxGetDigPatternRefTrigPattern`
- `DAQmxSetDigPatternRefTrigPattern`
- `DAQmxResetDigPatternRefTrigPattern`
Reference >> Digital Pattern >> Trigger When

**Data Type:** int32

**Description:** Specifies whether the Reference Trigger occurs when the physical channels specified with **Source** match or differ from the digital pattern specified with **Pattern**.

**Valid values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_PatternMatches</td>
<td>10254</td>
<td>Trigger when the physical channels match the specified pattern.</td>
</tr>
<tr>
<td>DAQmx_Val_PatternDoesNotMatch</td>
<td>10253</td>
<td>Trigger when the physical channels do not match the specified pattern.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDigPatternRefTrigWhen
- DAQmxSetDigPatternRefTrigWhen
- DAQmxResetDigPatternRefTrigWhen
Reference >> Analog Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Reference Trigger.

For E Series devices, if you use a channel name, the channel must be the only channel in the task. The only terminal you can use for E Series devices is PFI0.

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeRefTrigSrc`
- `DAQmxSetAnlgEdgeRefTrigSrc`
- `DAQmxResetAnlgEdgeRefTrigSrc`
Reference >> Analog Edge >> Slope

**Data Type:** int32

**Description:** Specifies on which slope of the source signal the Reference Trigger occurs.

### Valid values

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_RisingSlope</td>
<td>10280</td>
<td>Trigger on the rising slope of the signal.</td>
</tr>
<tr>
<td>DAQmx_Val_FallingSlope</td>
<td>10171</td>
<td>Trigger on the falling slope of the signal.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeRefTrigSlope`
- `DAQmxSetAnlgEdgeRefTrigSlope`
- `DAQmxResetAnlgEdgeRefTrigSlope`
Reference >> Analog Edge >> Level

**Data Type:** float64

**Description:** Specifies in the units of the measurement the threshold at which the Reference Trigger occurs. Use **Slope** to specify on which slope to trigger at this threshold.

You can get/set/reset this property using:

- DAQmxGetAnlgEdgeRefTrigLvl
- DAQmxSetAnlgEdgeRefTrigLvl
- DAQmxResetAnlgEdgeRefTrigLvl
Reference >> Analog Edge >> Hysteresis

**Data Type:** float64

**Description:** Specifies a hysteresis level in the units of the measurement. If `Slope` is DAQmx_Val_RisingSlope, the trigger does not deassert until the source signal passes below `Level` minus the hysteresis. If `Slope` is DAQmx_Val_FallingSlope, the trigger does not deassert until the source signal passes above `Level` plus the hysteresis.

You can get/set/reset this property using:

- `DAQmxGetAnlgEdgeRefTrigHyst`
- `DAQmxSetAnlgEdgeRefTrigHyst`
- `DAQmxResetAnlgEdgeRefTrigHyst`
Reference >> Analog Edge >> Coupling

**Data Type:** int32

**Description:** Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Alternating Current.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Direct Current.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAnlgEdgeRefTrigCoupling
- DAQmxSetAnlgEdgeRefTrigCoupling
- DAQmxResetAnlgEdgeRefTrigCoupling
Reference >> Analog Window >> Source

**Data Type:** char*

**Description:** Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Reference Trigger.

For E Series devices, if you use a channel name, the channel must be the only channel in the task. The only terminal you can use for E Series devices is PFI0.

You can get/set/reset this property using:

- DAQmxGetAnlgWinRefTrigSrc
- DAQmxSetAnlgWinRefTrigSrc
- DAQmxResetAnlgWinRefTrigSrc
Reference >> Analog Window >> Trigger When

**Data Type:** int32

**Description:** Specifies whether the Reference Trigger occurs when the source signal enters the window or when it leaves the window. Use **Bottom** and **Top** to specify the window.

**Valid values**

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_EnteringWin</td>
<td>10163</td>
<td>Trigger when the signal enters the window.</td>
</tr>
<tr>
<td>DAQmx_Val_LeavingWin</td>
<td>10208</td>
<td>Trigger when the signal leaves the window.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAnlgWinRefTrigWhen`
- `DAQmxSetAnlgWinRefTrigWhen`
- `DAQmxResetAnlgWinRefTrigWhen`
**Reference >> Analog Window >> Top**

**Data Type:** float64

**Description:** Specifies the upper limit of the window. Specify this value in the units of the measurement.

You can get/set/reset this property using:

- `DAQmxGetAnlgWinRefTrigTop`
- `DAQmxSetAnlgWinRefTrigTop`
- `DAQmxResetAnlgWinRefTrigTop`
Reference >> Analog Window >> Bottom

**Data Type:** float64

**Description:** Specifies the lower limit of the window. Specify this value in the units of the measurement.

You can get/set/reset this property using:

- `DAQmxGetAnlgWinRefTrigBtm`
- `DAQmxSetAnlgWinRefTrigBtm`
- `DAQmxResetAnlgWinRefTrigBtm`
Reference >> Analog Window >> Coupling

**Data Type:** int32

**Description:** Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Alternating Current.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Direct Current.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAnlgWinRefTrigCoupling
- DAQmxSetAnlgWinRefTrigCoupling
- DAQmxResetAnlgWinRefTrigCoupling
More >> Advance >> Trigger Type

Data Type: int32
Description: Specifies the type of trigger to use to advance to the next entry in a switch scan list.

Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DigEdge</td>
<td>10150</td>
<td>Advance to the next entry in a scan list on the rising or falling edge of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_Software</td>
<td>10292</td>
<td>Advance to the next entry in a scan list when you call DAQmxSendSoftwareTrigger().</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Advance through all entries in the scan list as fast as possible.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetAdvTrigType
- DAQmxSetAdvTrigType
- DAQmxResetAdvTrigType
More >> Advance >> Digital Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a terminal where there is a digital signal to use as the source of the Advance Trigger.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeAdvTrigSrc`
- `DAQmxSetDigEdgeAdvTrigSrc`
- `DAQmxResetDigEdgeAdvTrigSrc`
**More >> Advance >> Digital Edge >> Edge**

**Data Type:** int32

**Description:** Specifies on which edge of a digital signal to advance to the next entry in a scan list.

### Valid values

<table>
<thead>
<tr>
<th>Module</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetDigEdgeAdvTrigEdge`
- `DAQmxSetDigEdgeAdvTrigEdge`
- `DAQmxResetDigEdgeAdvTrigEdge`
More >> Advance >> Digital Edge >> Digital Filter >>
Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeAdvTrigDigFltrEnable`
- `DAQmxSetDigEdgeAdvTrigDigFltrEnable`
- `DAQmxResetDigEdgeAdvTrigDigFltrEnable`
More >> Handshake >> Trigger Type

**Data Type:** int32

**Description:** Specifies the type of Handshake Trigger to use.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_Interlocked</th>
<th>12549</th>
<th>Use the Handshake Trigger as a control signal for asynchronous handshaking, such as 8255 handshaking.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Start the measurement or generation immediately when you start the task.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetHshkTrigType`
- `DAQmxSetHshkTrigType`
- `DAQmxResetHshkTrigType`
More >> Handshake >> Interlocked >> Source

**Data Type:** char*

**Description:** Specifies the source terminal of the Handshake Trigger.

You can get/set/reset this property using:

- DAQmxGetInterlockedHshkTrigSrc
- DAQmxSetInterlockedHshkTrigSrc
- DAQmxResetInterlockedHshkTrigSrc
More >> Handshake >> Interlocked >> Asserted Level

**Data Type:** int32

**Description:** Specifies the asserted level of the Handshake Trigger.

**Valid values**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>High state.</td>
<td></td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Low state.</td>
<td></td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetInterlockedHshkTrigAssertedLvl`
- `DAQmxSetInterlockedHshkTrigAssertedLvl`
- `DAQmxResetInterlockedHshkTrigAssertedLvl`
**More >> Pause >> Trigger Type**

**Data Type:** int32  
**Description:** Specifies the type of trigger to use to pause a task.

### Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_AnlgLvl</th>
<th>10101</th>
<th>Pause the measurement or generation while an analog signal is above or below a level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AnlgWin</td>
<td>10103</td>
<td>Pause the measurement or generation while an analog signal is either inside or outside of a range of values.</td>
</tr>
<tr>
<td>DAQmx_Val_DigLvl</td>
<td>10152</td>
<td>Pause the measurement or generation while a digital signal is at either a high or low state.</td>
</tr>
<tr>
<td>DAQmx_Val_DigPattern</td>
<td>10398</td>
<td>Pause the measurement or generation while digital physical channels either match or do not match a digital pattern.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Do not pause the measurement or generation.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetPauseTrigType
- DAQmxSetPauseTrigType
- DAQmxResetPauseTrigType
Data Type: char*

Description: Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.

For E Series devices, if you use a channel name, the channel must be the only channel in the task. The only terminal you can use for E Series devices is PFI0.

You can get/set/reset this property using:

- DAQmxGetAnlgLvPauseTrigSrc
- DAQmxSetAnlgLvPauseTrigSrc
- DAQmxResetAnlgLvPauseTrigSrc
More >> Pause >> Analog Level >> Pause When

Data Type: int32
Description: Specifies whether the task pauses above or below the threshold you specify with Level.

Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AboveLvl 10093</td>
<td>Pause the measurement or generation while the signal is above the threshold.</td>
</tr>
<tr>
<td>DAQmx_Val_BelowLvl 10107</td>
<td>Pause the measurement or generation while the signal is below the threshold.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAnlgLvlPauseTrigWhen
DAQmxSetAnlgLvlPauseTrigWhen
DAQmxResetAnlgLvlPauseTrigWhen
More >> Pause >> Analog Level >> Level

**Data Type:** float64

**Description:** Specifies the threshold at which to pause the task. Specify this value in the units of the measurement or generation. Use [Pause When](#) to specify whether the task pauses above or below this threshold.

You can get/set/reset this property using:

- `DAQmxGetAnlgLvlPauseTrigLvl`
- `DAQmxSetAnlgLvlPauseTrigLvl`
- `DAQmxResetAnlgLvlPauseTrigLvl`
More >> Pause >> Analog Level >> Hysteresis

Data Type: float64

Description: Specifies a hysteresis level in the units of the measurement or generation. If Pause When is DAQmx_Val_AboveLvl, the trigger does not deassert until the source signal passes below Level minus the hysteresis. If Pause When is DAQmx_Val_BelowLvl, the trigger does not deassert until the source signal passes above Level plus the hysteresis.

You can get/set/reset this property using:

- `DAQmxGetAnlgLvlPauseTrigHyst`
- `DAQmxSetAnlgLvlPauseTrigHyst`
- `DAQmxResetAnlgLvlPauseTrigHyst`
**More >> Pause >> Analog Level >> Coupling**

**Data Type:** int32

**Description:** Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Alternating Current.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Direct Current.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAnlgLvlPauseTrigCoupling`
- `DAQmxSetAnlgLvlPauseTrigCoupling`
- `DAQmxResetAnlgLvlPauseTrigCoupling`
Data Type: char*

Description: Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.

For E Series devices, if you use a channel name, the channel must be the only channel in the task. The only terminal you can use for E Series devices is PFI0.

You can get/set/reset this property using:

DAQmxGetAnlgWinPauseTrigSrc
DAQmxSetAnlgWinPauseTrigSrc
DAQmxResetAnlgWinPauseTrigSrc
More >> Pause >> Analog Window >> Pause When

**Data Type:** int32

**Description:** Specifies whether the task pauses while the trigger signal is inside or outside the window you specify with Bottom and Top.

<table>
<thead>
<tr>
<th>Valid values</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_InsideWin</td>
<td>10199</td>
<td>Pause the measurement or generation while the trigger is inside the window.</td>
</tr>
<tr>
<td>DAQmx_Val_OutsideWin</td>
<td>10251</td>
<td>Pause the measurement or generation while the signal is outside the window.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetAnlgWinPauseTrigWhen`
- `DAQmxSetAnlgWinPauseTrigWhen`
- `DAQmxResetAnlgWinPauseTrigWhen`
More >> Pause >> Analog Window >> Top

**Data Type:** float64

**Description:** Specifies the upper limit of the window. Specify this value in the units of the measurement or generation.

You can get/set/reset this property using:

- DAQmxGetAnlgWinPauseTrigTop
- DAQmxSetAnlgWinPauseTrigTop
- DAQmxResetAnlgWinPauseTrigTop
Data Type: float64

Description: Specifies the lower limit of the window. Specify this value in the units of the measurement or generation.

You can get/set/reset this property using:

DAQmxGetAnlgWinPauseTrigBtm
DAQmxSetAnlgWinPauseTrigBtm
DAQmxResetAnlgWinPauseTrigBtm
More >> Pause >> Analog Window >> Coupling

Data Type: int32

Description: Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_AC</th>
<th>10045</th>
<th>Alternating Current.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DC</td>
<td>10050</td>
<td>Direct Current.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetAnlgWinPauseTrigCoupling
DAQmxSetAnlgWinPauseTrigCoupling
DAQmxResetAnlgWinPauseTrigCoupling
Data Type: char*

Description: Specifies the name of a terminal where there is a digital signal to use as the source of the Pause Trigger.

You can get/set/reset this property using:

DAQmxGetDigLvlPauseTrigSrc
DAQmxSetDigLvlPauseTrigSrc
DAQmxResetDigLvlPauseTrigSrc
**More >> Pause >> Digital Level >> Pause When**

- **Data Type:** int32
- **Description:** Specifies whether the task pauses while the signal is high or low.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DAQmx_Val_High</code></td>
<td>10192</td>
<td>High state.</td>
</tr>
<tr>
<td><code>DAQmx_Val_Low</code></td>
<td>10214</td>
<td>Low state.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetDigLvlPauseTrigWhen`
- `DAQmxSetDigLvlPauseTrigWhen`
- `DAQmxResetDigLvlPauseTrigWhen`
More >> Pause >> Digital Level >> Digital Filter >> Enable

**Data Type:** bool32

**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `DAQmxGetDigLvPauseTrigDigFltrEnable`
- `DAQmxSetDigLvPauseTrigDigFltrEnable`
- `DAQmxResetDigLvPauseTrigDigFltrEnable`
More >> Pause >> Digital Level >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetDigLvlPauseTrigDigFltrMinPulseWidth`
- `DAQmxSetDigLvlPauseTrigDigFltrMinPulseWidth`
- `DAQmxResetDigLvlPauseTrigDigFltrMinPulseWidth`
More >> Pause >> Digital Level >> Digital Filter >> Timebase >> Source

**Data Type:** char

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetDigLvlPauseTrigDigFltrTimebaseSrc`
- `DAQmxSetDigLvlPauseTrigDigFltrTimebaseSrc`
- `DAQmxResetDigLvlPauseTrigDigFltrTimebaseSrc`
More >> Pause >> Digital Level >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- `DAQmxGetDigLvlPauseTrigDigFltrTimebaseRate`
- `DAQmxSetDigLvlPauseTrigDigFltrTimebaseRate`
- `DAQmxResetDigLvlPauseTrigDigFltrTimebaseRate`
More >> Pause >> Digital Level >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- `DAQmxGetDigLvlPauseTrigDigSyncEnable`
- `DAQmxSetDigLvlPauseTrigDigSyncEnable`
- `DAQmxResetDigLvlPauseTrigDigSyncEnable`
More >> Pause >> Digital Pattern >> Source

**Data Type:** char*

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetDigPatternPauseTrigSrc`
- `DAQmxSetDigPatternPauseTrigSrc`
- `DAQmxResetDigPatternPauseTrigSrc`
More >> Pause >> Digital Pattern >> Pattern

**Data Type:** char*

**Description:** Specifies the digital pattern that must be met for the Pause Trigger to occur.

You can get/set/reset this property using:

- DAQmxGetDigPatternPauseTrigPattern
- DAQmxSetDigPatternPauseTrigPattern
- DAQmxResetDigPatternPauseTrigPattern
Data Type: int32

Description: Specifies if the Pause Trigger occurs when the physical channels specified with Source match or differ from the digital pattern specified with Pattern.

Valid values

| DAQmx_Val_PatternMatches   | 10254 | Trigger when the physical channels match the specified pattern. |
| DAQmx_Val_PatternDoesNotMatch | 10253 | Trigger when the physical channels do not match the specified pattern. |

You can get/set/reset this property using:

- DAQmxGetDigPatternPauseTrigWhen
- DAQmxSetDigPatternPauseTrigWhen
- DAQmxResetDigPatternPauseTrigWhen
**More >> Arm Start >> Trigger Type**

**Data Type:** int32

**Description:** 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.

**Valid values**

<table>
<thead>
<tr>
<th>DAQmx_Val_DigEdge</th>
<th>10150</th>
<th>Trigger on a rising or falling edge of a digital signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Disable the trigger.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetArmStartTrigType
- DAQmxSetArmStartTrigType
- DAQmxResetArmStartTrigType
More >> Arm Start >> Digital Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a terminal where there is a digital signal to use as the source of the Arm Start Trigger.

You can get/set/reset this property using:

`DAQmxGetDigEdgeArmStartTrigSrc`

`DAQmxSetDigEdgeArmStartTrigSrc`

`DAQmxResetDigEdgeArmStartTrigSrc`
More >> Arm Start >> Digital Edge >> Edge

**Data Type:** int32

**Description:** Specifies on which edge of a digital signal to arm the task for a Start Trigger.

**Valid values**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetDigEdgeArmStartTrigEdge
- DAQmxSetDigEdgeArmStartTrigEdge
- DAQmxResetDigEdgeArmStartTrigEdge
**More >> Arm Start >> Digital Edge >> Digital Filter >> Enable**

**Data Type:** bool32  
**Description:** Specifies whether to apply the pulse width filter to the signal.

You can get/set/reset this property using:

- `$DAQmxGetDigEdgeArmStartTrigDigFltrEnable`  
- `$DAQmxSetDigEdgeArmStartTrigDigFltrEnable`  
- `$DAQmxResetDigEdgeArmStartTrigDigFltrEnable`
More >> Arm Start >> Digital Edge >> Digital Filter >> Minimum Pulse Width

**Data Type:** float64

**Description:** Specifies in seconds the minimum pulse width the filter recognizes.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeArmStartTrigDigFltrMinPulseWidth`
- `DAQmxSetDigEdgeArmStartTrigDigFltrMinPulseWidth`
- `DAQmxResetDigEdgeArmStartTrigDigFltrMinPulseWidth`
More >> Arm Start >> Digital Edge >> Digital Filter >> Timebase >> Source

**Data Type:** char*

**Description:** Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

You can get/set/reset this property using:

- `DAQmxGetDigEdgeArmStartTrigDigFltrTimebaseSrc`
- `DAQmxSetDigEdgeArmStartTrigDigFltrTimebaseSrc`
- `DAQmxResetDigEdgeArmStartTrigDigFltrTimebaseSrc`
More >> Arm Start >> Digital Edge >> Digital Filter >> Timebase >> Rate

**Data Type:** float64

**Description:** Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

You can get/set/reset this property using:

- DAQmxGetDigEdgeArmStartTrigDigFiltTimebaseRate
- DAQmxSetDigEdgeArmStartTrigDigFiltTimebaseRate
- DAQmxResetDigEdgeArmStartTrigDigFiltTimebaseRate
More >> Arm Start >> Digital Edge >> Digital Synchronization >> Enable

**Data Type:** bool32

**Description:** Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

This property does not affect the minimum pulse width recognized by the device, but setting this property to TRUE does limit the speed at which the device recognizes transitions to less than the frequency of the internal timebase.

You can get/set/reset this property using:

- DAQmxGetDigEdgeArmStartTrigDigSyncEnable
- DAQmxSetDigEdgeArmStartTrigDigSyncEnable
- DAQmxResetDigEdgeArmStartTrigDigSyncEnable
**Timeout**

**Data Type:** float64

**Description:** 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.

You can get/set/reset this property using:

- `DAQmxGetWatchdogTimeout`
- `DAQmxSetWatchdogTimeout`
- `DAQmxResetWatchdogTimeout`
Expiration Trigger >> Trigger Type

**Data Type:** int32

**Description:** Specifies the type of trigger to use to expire a watchdog task.

### Valid values

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_DigEdge</td>
<td>10150</td>
<td>Trigger on a rising or falling edge of a digital signal.</td>
</tr>
<tr>
<td>DAQmx_Val_None</td>
<td>10230</td>
<td>Disable the trigger.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetWatchdogExpirTrigType`
- `DAQmxSetWatchdogExpirTrigType`
- `DAQmxResetWatchdogExpirTrigType`
Expiration Trigger >> Digital Edge >> Source

**Data Type:** char*

**Description:** Specifies the name of a terminal where a digital signal exists to use as the source of the Expiration Trigger.

You can get/set/reset this property using:

`DAQmxGetDigEdgeWatchdogExpirTrigSrc`

`DAQmxSetDigEdgeWatchdogExpirTrigSrc`

`DAQmxResetDigEdgeWatchdogExpirTrigSrc`
Expiration Trigger >> Digital Edge >> Edge

**Data Type:** int32

**Description:** Specifies on which edge of a digital signal to expire the watchdog task.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Rising</td>
<td>10280</td>
<td>Rising edge(s).</td>
</tr>
<tr>
<td>DAQmx_Val_Falling</td>
<td>10171</td>
<td>Falling edge(s).</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetDigEdgeWatchdogExpirTrigEdge`
- `DAQmxSetDigEdgeWatchdogExpirTrigEdge`
- `DAQmxResetDigEdgeWatchdogExpirTrigEdge`
## Expiration States >> Digital Output >> Expiration State

**Data Type:** int32  
**Description:** 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.

### Valid values

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_High</td>
<td>10192</td>
<td>Logic high.</td>
</tr>
<tr>
<td>DAQmx_Val_Low</td>
<td>10214</td>
<td>Logic low.</td>
</tr>
<tr>
<td>DAQmx_Val_Tristate</td>
<td>10310</td>
<td>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.</td>
</tr>
<tr>
<td>DAQmx_Val_NoChange</td>
<td>10160</td>
<td>Do not change the state of the lines. On some devices, you can select this value only for entire ports.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- `DAQmxGetWatchdogDOExpirState`
- `DAQmxSetWatchdogDOExpirState`
- `DAQmxResetWatchdogDOExpirState`
**Status >> Expired**

**Data Type:** bool32  
**Description:** Indicates if the watchdog timer expired. You can read this property only while the task is running.  
**Restrictions:** Not Settable

You can get this property using:  

[DAQmxGetWatchdogHasExpired](#)
Relative To

Data Type: int32

Description: 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.

Valid values

<table>
<thead>
<tr>
<th>DAQmx_Val_FirstSample</th>
<th>10424</th>
<th>Write samples relative to the first sample.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_CurrWritePos</td>
<td>10430</td>
<td>Write samples relative to the current position in the buffer.</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

DAQmxGetWriteRelativeTo
DAQmxSetWriteRelativeTo
DAQmxResetWriteRelativeTo
Offset

**Data Type:** int32

**Description:** 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**.

You can get/set/reset this property using:

- `DAQmxGetWriteOffset`
- `DAQmxSetWriteOffset`
- `DAQmxResetWriteOffset`
Regeneration Mode

**Data Type:** int32

**Description:** Specifies whether to allow NI-DAQmx to generate the same data multiple times.

If you enable regeneration and write new data to the buffer, NI-DAQmx can generate a combination of old and new data, a phenomenon called **glitching**.

### Valid values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_AllowRegen</td>
<td>10097</td>
</tr>
<tr>
<td>DAQmx_Val_DoNotAllowRegen</td>
<td>10158</td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetWriteRegenMode
- DAQmxSetWriteRegenMode
- DAQmxResetWriteRegenMode
**Status >> Current Write Position**

**Data Type:** uInt64

**Description:** Indicates the position in the buffer of the next sample to generate. This value is identical for all channels in the task.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetWriteCurrWritePos](#)
**Status >> Overcurrent >> Overcurrent Channels Exist**

**Data Type:** bool32

**Description:** 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 Channels. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetWriteOvercurrentChansExist`
**Status >> Overcurrent >> Overcurrent Channels**

**Data Type:** char*

**Description:** Indicates the names of any virtual channels in the task for which an overcurrent condition has been detected. You must read Overcurrent Channels Exist before you read this property. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetWriteOvercurrentChans
Status >> Open Current Loop >> Open Current Loop Channels Exist

Data Type: bool32

Description: 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 Channels. Otherwise, you will receive an error.

Restrictions: Not Settable

You can get this property using:

DAQmxGetWriteOpenCurrentLoopChansExist
**Status >> Open Current Loop >> Open Current Loop Channels**

**Data Type:** char*

**Description:** Indicates the names of any virtual channels in the task for which the device(s) detected an open current loop. You must read `Open Current Loop Channels Exist` before you read this property. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetWriteOpenCurrentLoopChans`
**Status >> Power Supply Fault >> Power Supply Fault Channels Exist**

**Data Type:** bool32

**Description:** 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 Channels](#). Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

`DAQmxGetWritePowerSupplyFaultChansExist`
Status >> Power Supply Fault >> Power Supply Fault Channels

**Data Type:** char*

**Description:** Indicates the names of any virtual channels in the task that have a power supply fault. You must read Power Supply Fault Channels Exist before you read this property. Otherwise, you will receive an error.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetWritePowerSupplyFaultChans
**Status >> Space Available in Buffer**

**Data Type:** uInt32

**Description:** Indicates in samples per channel the amount of available space in the buffer.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetWriteSpaceAvail](https://www.ni.com)
Status >> Total Samples Per Channel Generated

**Data Type:** uInt64

**Description:** Indicates the total number of samples generated by each channel in the task. This value is identical for all channels in the task.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetWriteTotalSampPerChanGenerated
Advanced >> Raw Data Width

**Data Type:** uint32

**Description:** Indicates in bytes the required size of a raw sample to write to the task.

**Restrictions:** Not Settable

You can get this property using:

**Advanced >> Number of Channels**

**Data Type:**  uInt32

**Description:** Indicates the number of channels that an NI-DAQmx Write function writes to the task. This value is the number of channels in the task.

**Restrictions:** Not Settable

You can get this property using:

DAQmxGetWriteNumChans
Advanced >> Wait Mode

Data Type: int32

Description: Specifies how an NI-DAQmx Write function waits for space to become available in the buffer.

<table>
<thead>
<tr>
<th>Valid values</th>
<th>12524</th>
<th>12525</th>
<th>12547</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQmx_Val_Poll</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Repeatedly check for available buffer space as fast as possible. This mode allows for the highest sampling rates at the expense of CPU efficiency.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAQmx_Val_Yield</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Repeatedly check for available buffer space, but yield control to other threads after each check. This mode offers a balance between sampling rate and CPU efficiency.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAQmx_Val_Sleep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check for available buffer space once per the amount of time specified in Sleep Time.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can get/set/reset this property using:

- DAQmxGetWriteWaitMode
- DAQmxSetWriteWaitMode
- DAQmxResetWriteWaitMode
Advanced >> Sleep Time

**Data Type:** float64

**Description:** Specifies in seconds the amount of time to sleep after checking for available buffer space if **Wait Mode** is DAQmx_Val_Sleep.

You can get/set/reset this property using:

- DAQmxGetWriteSleepTime
- DAQmxSetWriteSleepTime
- DAQmxResetWriteSleepTime
Advanced >> Next Write Is Last

**Data Type:** bool32

**Description:** Specifies that the next samples written are the last samples you want to generate. Use this property when performing continuous generation to prevent underflow errors after writing the last sample. **Regeneration Mode** must be DAQmx_Val_DoNotAllowRegen to use this property.

You can get/set/reset this property using:

- DAQmxGetWriteNextWriteIsLast
- DAQmxSetWriteNextWriteIsLast
- DAQmxResetWriteNextWriteIsLast
Advanced >> Digital Output >> Number of Booleans Per Channel

**Data Type:** uInt32

**Description:** Indicates the number of bytes expected per channel in a sample for line-based writes. If a channel has fewer lines than this number, NI-DAQmx ignores the extra bytes.

**Restrictions:** Not Settable

You can get this property using:

[DAQmxGetWriteDigitalLinesBytesPerChan](http://example.com/DAQmxGetWriteDigitalLinesBytesPerChan)
Get/Set/Reset Buf_Input_BufSize

int32 __CFUNC DAQmxGetBufInputBufSize(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetBufInputBufSize(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetBufInputBufSize(TaskHandle taskHandle);
**Purpose**

DAQmxGetBufInputBufSize gets the **Input > Buffer Size** property.

DAQmxSetBufInputBufSize sets the **Input > Buffer Size** property.

DAQmxResetBufInputBufSize resets the **Input > Buffer Size** property.
Get/Set/Reset Buf_Input_OnbrdBufSize

int32 __CFUNC DAQmxGetBufInputOnbrdBufSize(TaskHandle taskHandle, uInt32 *data);
**Purpose**

DAQmxGetBufInputOnbrdBufSize gets the [Input >> Onboard Buffer Size](#) property.
Get/Set/Reset Buf_Output_BufSize

int32 __CFUNC DAQmxGetBufOutputBufSize(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetBufOutputBufSize(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetBufOutputBufSize(TaskHandle taskHandle);
Purpose

DAQmxGetBufOutputBufSize gets the Output >> Buffer Size property.
DAQmxSetBufOutputBufSize sets the Output >> Buffer Size property.
DAQmxResetBufOutputBufSize resets the Output >> Buffer Size property.
Get/Set/Reset Buf_Output_OnbrdBufSize

int32 __CFUNC DAQmxGetBufOutputOnbrdBufSize(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetBufOutputOnbrdBufSize(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetBufOutputOnbrdBufSize(TaskHandle taskHandle);
Purpose

DAQmxGetBufOutputOnbrdBufSize gets the Output >> Onboard Buffer Size property.
DAQmxSetBufOutputOnbrdBufSize sets the Output >> Onboard Buffer Size property.
DAQmxResetBufOutputOnbrdBufSize resets the Output >> Onboard Buffer Size property.
Get/Set/Reset SelfCal_Supported

int32 __CFUNC DAQmxGetSelfCalSupported(const char deviceName[], bool32 *data);
Purpose

DAQmxGetSelfCalSupported gets the Self Calibration >> Is Supported property.
Get/Set/Reset SelfCal_LastTemp

int32 __CFUNC DAQmxGetSelfCalLastTemp(const char deviceName[], float64 *data);
Purpose

DAQmxGetSelfCalLastTemp gets the Self Calibration >> Last Self Calibration Temperature property.
Get/Set/Reset ExtCal_RecommendedInterval

int32 __CFUNC DAQmxGetExtCalRecommendedInterval(const char *deviceName[], Uint32 *data);
Purpose

DAQmxGetExtCalRecommendedInterval gets the External Calibration >> Recommended Interval property.
Get/Set/Reset ExtCal_LastTemp

int32 __CFUNC DAQmxGetExtCalLastTemp(const char deviceName[], float64 *data);
Purpose

DAQmxGetExtCalLastTemp gets the External Calibration >> Last External Calibration Temperature property.
**Get/Set/Reset Cal_UserDefinedInfo**

```c
int32 __CFUNC DAQmxGetCalUserDefinedInfo(const char deviceName[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCalUserDefinedInfo(const char deviceName[], const char *data);
```
**Purpose**

DAQmxGetCalUserDefinedInfo gets the User-Defined Information property.
DAQmxSetCalUserDefinedInfo sets the User-Defined Information property.
Get/Set/Reset Cal_UserDefinedInfo_MaxSize

int32 __CFUNC DAQmxGetCalUserDefinedInfoMaxSize(const char deviceName[], uInt32 *data);
Purpose

DAQmxGetCalUserDefinedInfoMaxSize gets the `User-Defined Information >> Max Size` property.
Get/Set/Reset Cal_DevTemp

int32 __CFUNC DAQmxGetCalDevTemp(const char deviceName[], float64 *data);
Purpose

DAQmxGetCalDevTemp gets the More >> Device Temperature property.
Get/Set/Reset AI_Max

int32 __CFUNC DAQmxGetAIMax(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIMax(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIMax(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIMax gets the Analog Input >> Maximum Value property.
DAQmxSetAIMax sets the Analog Input >> Maximum Value property.
DAQmxResetAIMax resets the Analog Input >> Maximum Value property.
Get/Set/Reset AI_Min

int32 __CFUNC DAQmxGetAIMin(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIMin(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIMin(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIMin gets the Analog Input >> Minimum Value property.
DAQmxSetAIMin sets the Analog Input >> Minimum Value property.
DAQmxResetAIMin resets the Analog Input >> Minimum Value property.
Get/Set/Reset AI_CustomScaleName

int32 __CFUNC DAQmxGetAI_CustomScaleName(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAI_CustomScaleName(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAI_CustomScaleName(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAICustomScaleName gets the Analog Input >> Custom Scale Name property.

DAQmxSetAICustomScaleName sets the Analog Input >> Custom Scale Name property.

DAQmxResetAICustomScaleName resets the Analog Input >> Custom Scale Name property.
Get/Set/Reset AI_MeasType

int32 __CFUNC DAQmxGetAIMeasType(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetAIMeasType gets the Analog Input >> Measurement Type property.
**Get/Set/Reset AI_Voltage_Units**

```c
int32 __CFUNC DAQmxGetAIVoltageUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIVoltageUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIVoltageUnits(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetAIVoltageUnits gets the [Analog Input >> Voltage >> Units](#) property.

DAQmxSetAIVoltageUnits sets the [Analog Input >> Voltage >> Units](#) property.

DAQmxResetAIVoltageUnits resets the [Analog Input >> Voltage >> Units](#) property.
Get/Set/Reset AI_Voltage_dBRef

int32 __CFUNC DAQmxGetAIVoltagedBRef(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIVoltagedBRef(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIVoltagedBRef(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIVoltagedBRef gets the Analog Input >> Voltage >> dB Reference property.
DAQmxSetAIVoltagedBRef sets the Analog Input >> Voltage >> dB Reference property.
DAQmxResetAIVoltagedBRef resets the Analog Input >> Voltage >> dB Reference property.
Get/Set/Reset AI_Voltage_ACRMS_Units

int32 __CFUNC DAQmxGetAIVoltageACRMSUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIVoltageACRMSUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIVoltageACRMSUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIVoltageACRMSUnits gets the Analog Input >> Voltage >> AC RMS Voltage >> Units property.

DAQmxSetAIVoltageACRMSUnits sets the Analog Input >> Voltage >> AC RMS Voltage >> Units property.

DAQmxResetAIVoltageACRMSUnits resets the Analog Input >> Voltage >> AC RMS Voltage >> Units property.
Get/Set/Reset AI_Temp_Units

int32 __CFUNC DAQmxGetAITempUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAITempUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAITempUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAITempUnits gets the Analog Input >> Temperature >> Units property.
DAQmxSetAITempUnits sets the Analog Input >> Temperature >> Units property.
DAQmxResetAITempUnits resets the Analog Input >> Temperature >> Units property.
Get/Set/Reset AI_Thrmcpl_Type

int32 __CFUNC DAQmxGetAIThrmcplType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIThrmcplType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIThrmcplType(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThrmcplType gets the Analog Input >> Temperature >> Thermocouple >> Type property.

DAQmxSetAIThrmcplType sets the Analog Input >> Temperature >> Thermocouple >> Type property.

DAQmxResetAIThrmcplType resets the Analog Input >> Temperature >> Thermocouple >> Type property.
Get/Set/Reset AI_Thrmcpl_ScaleType

int32 __CFUNC DAQmxGetAIThrmcplScaleType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIThrmcplScaleType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIThrmcplScaleType(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThrmcplScaleType gets the `Analog Input >> Temperature >> Thermocouple >> Scale Type` property.

DAQmxSetAIThrmcplScaleType sets the `Analog Input >> Temperature >> Thermocouple >> Scale Type` property.

DAQmxResetAIThrmcplScaleType resets the `Analog Input >> Temperature >> Thermocouple >> Scale Type` property.
Get/Set/Reset AI_Thrmcpl_CJCSrc

int32 __CFUNC DAQmxGetAIThrmcplCJCSrc(TaskHandle taskHandle, const char channel[], int32 *data);
**Purpose**

DAQmxGetAIThrmcplCJCSrc gets the Analog Input >> Temperature >> Thermocouple >> CJC Source property.
Get/Set/Reset AI_Thrmcpl_CJCVa

int32 __CFUNC DAQmxGetAIThrmcplCJCVa(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIThrmcplCJCVa(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIThrmcplCJCVa(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThrmcplCJCVal gets the Analog Input >> Temperature >> Thermocouple >> CJC Value property.

DAQmxSetAIThrmcplCJCVal sets the Analog Input >> Temperature >> Thermocouple >> CJC Value property.

DAQmxResetAIThrmcplCJCVal resets the Analog Input >> Temperature >> Thermocouple >> CJC Value property.
Get/Set/Reset AI_Thrmcpl_CJCChan

int32 __CFUNC DAQmxGetAIThrmcplCJCChan(TaskHandle taskHandle,
const char channel[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetAIThrmcplCJCChan gets the Analog Input >> Temperature >> Thermocouple >> CJC Channel property.
Get/Set/Reset AI_RTD_Type

```c
int32 __CFUNC DAQmxGetAIRTDType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIRTDType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIRTDType(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAIRTDType gets the Analog Input >> Temperature >> RTD >> Type property.
DAQmxSetAIRTDType sets the Analog Input >> Temperature >> RTD >> Type property.
DAQmxResetAIRTDType resets the Analog Input >> Temperature >> RTD >> Type property.
Get/Set/Reset AI_RTD_R0

int32 __CFUNC DAQmxGetAIRTDR0(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIRTDR0(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIRTDR0(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIRTDR0 gets the Analog Input >> Temperature >> RTD >> R0 property.
DAQmxSetAIRTDR0 sets the Analog Input >> Temperature >> RTD >> R0 property.
DAQmxResetAIRTDR0 resets the Analog Input >> Temperature >> RTD >> R0 property.
Get/Set/Reset AI_RTD_A

int32 __CFUNC DAQmxGetAIRTDA(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIRTDA(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIRTDA(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIRTDA gets the Analog Input >> Temperature >> RTD >> Custom >> A property.
DAQmxSetAIRTDA sets the Analog Input >> Temperature >> RTD >> Custom >> A property.
DAQmxResetAIRTDA resets the Analog Input >> Temperature >> RTD >> Custom >> A property.
Get/Set/Reset AI_RTD_B

int32 __CFUNC DAQmxGetAIWrite(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIWrite(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIWrite(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAI RTDB gets the Analog Input >> Temperature >> RTD >> Custom >> B property.

DAQmxSetAI RTDB sets the Analog Input >> Temperature >> RTD >> Custom >> B property.

DAQmxResetAI RTDB resets the Analog Input >> Temperature >> RTD >> Custom >> B property.
Get/Set/Reset AI_RTD_C

int32 __CFUNC DAQmxGetAIRTDC(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIRTDC(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIRTDC(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIRTDC gets the [Analog Input >> Temperature >> RTD >> Custom >> C] property.

DAQmxSetAIRTDC sets the [Analog Input >> Temperature >> RTD >> Custom >> C] property.

DAQmxResetAIRTDC resets the [Analog Input >> Temperature >> RTD >> Custom >> C] property.
Get/Set/Reset AI_Thrmstr_A

int32 __CFUNC DAQmxGetAIThrmstrA(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIThrmstrA(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIThrmstrA(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThmstrA gets the Analog Input >> Temperature >> Thermistor >> A property.

DAQmxSetAIThmstrA sets the Analog Input >> Temperature >> Thermistor >> A property.

DAQmxResetAIThmstrA resets the Analog Input >> Temperature >> Thermistor >> A property.
Get/Set/Reset AI_Thrmstr_B

int32 __CFUNC DAQmxGetAIThrmstrB(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIThrmstrB(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIThrmstrB(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThmstrB gets the Analog Input >> Temperature >> Thermistor >> B property.
DAQmxSetAIThmstrB sets the Analog Input >> Temperature >> Thermistor >> B property.
DAQmxResetAIThmstrB resets the Analog Input >> Temperature >> Thermistor >> B property.
Get/Set/Reset AI_Thrmstr_C

int32 __CFUNC DAQmxGetAIThmstrC(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIThmstrC(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIThmstrC(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThmstrC gets the Analog Input >> Temperature >> Thermistor >> C property.
DAQmxSetAIThmstrC sets the Analog Input >> Temperature >> Thermistor >> C property.
DAQmxResetAIThmstrC resets the Analog Input >> Temperature >> Thermistor >> C property.
Get/Set/Reset AI_Thrmstr_R1

int32 __CFUNC DAQmxGetAIThrmstrR1(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIThrmstrR1(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIThrmstrR1(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIThmstrR1 gets the Analog Input >> Temperature >> Thermistor >> R1 property.

DAQmxSetAIThmstrR1 sets the Analog Input >> Temperature >> Thermistor >> R1 property.

DAQmxResetAIThmstrR1 resets the Analog Input >> Temperature >> Thermistor >> R1 property.
Get/Set/Reset AI_ForceReadFromChan

int32 __CFUNC DAQmxGetAIForceReadFromChan(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIForceReadFromChan(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIForceReadFromChan(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIForceReadFromChan gets the Analog Input >> Temperature >> Advanced >> Force Read from Channel property.

DAQmxSetAIForceReadFromChan sets the Analog Input >> Temperature >> Advanced >> Force Read from Channel property.

DAQmxResetAIForceReadFromChan resets the Analog Input >> Temperature >> Advanced >> Force Read from Channel property.
Get/Set/Reset AI_Current_Units

int32 __CFUNC DAQmxGetAICurrentUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAICurrentUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAICurrentUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAICurrentUnits gets the Analog Input >> Current >> Units property. DAQmxSetAICurrentUnits sets the Analog Input >> Current >> Units property. DAQmxResetAICurrentUnits resets the Analog Input >> Current >> Units property.
Get/Set/Reset AI_Current_ACRMS_Units

int32 __CFUNC DAQmxGetAICurrentACRMSUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAICurrentACRMSUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAICurrentACRMSUnits(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAICurrentACRMSUnits gets the Analog Input >> Current >> AC RMS Current >> Units property.

DAQmxSetAICurrentACRMSUnits sets the Analog Input >> Current >> AC RMS Current >> Units property.

DAQmxResetAICurrentACRMSUnits resets the Analog Input >> Current >> AC RMS Current >> Units property.
Get/Set/Reset AI_Strain_Units

int32 __CFUNC DAQmxGetAIStrainUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIStrainUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIStrainUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIStrainUnits gets the Analog Input >> Strain >> Units property.
DAQmxSetAIStrainUnits sets the Analog Input >> Strain >> Units property.
DAQmxResetAIStrainUnits resets the Analog Input >> Strain >> Units property.
Get/Set/Reset AI_StrainGage_GageFactor

int32 __CFUNC DAQmxGetAIStainGageGageFactor(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIStainGageGageFactor(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIStainGageGageFactor(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIStrainGageGageFactor gets the Analog Input >> Strain >> Strain Gage >> Gage Factor property.

DAQmxSetAIStrainGageGageFactor sets the Analog Input >> Strain >> Strain Gage >> Gage Factor property.

DAQmxResetAIStrainGageGageFactor resets the Analog Input >> Strain >> Strain Gage >> Gage Factor property.
Get/Set/Reset AI_StrainGage_PoissonRatio

int32 __CFUNC DAQmxGetAIStainGagePoissonRatio(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIStainGagePoissonRatio(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIStainGagePoissonRatio(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIStrainGagePoissonRatio gets the Analog Input >> Strain >> Strain Gage >> Poisson Ratio property.

DAQmxSetAIStrainGagePoissonRatio sets the Analog Input >> Strain >> Strain Gage >> Poisson Ratio property.

DAQmxResetAIStrainGagePoissonRatio resets the Analog Input >> Strain >> Strain Gage >> Poisson Ratio property.
Get/Set/Reset AI_StrainGage_Cfg

int32 __CFUNC DAQmxGetAIStrainGageCfg(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIStrainGageCfg(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIStainGageCfg(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIStrainGageCfg gets the Analog Input >> Strain >> Strain Gage >> Configuration property.
DAQmxSetAIStrainGageCfg sets the Analog Input >> Strain >> Strain Gage >> Configuration property.
DAQmxResetAIStrainGageCfg resets the Analog Input >> Strain >> Strain Gage >> Configuration property.
Get/Set/Reset AI Resistance Units

int32 __CFUNC DAQmxGetAIResistanceUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIResistanceUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIResistanceUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIResistanceUnits gets the Analog Input >> Resistance >> Units property.
DAQmxSetAIResistanceUnits sets the Analog Input >> Resistance >> Units property.
DAQmxResetAIResistanceUnits resets the Analog Input >> Resistance >> Units property.
Get/Set/Reset AI_Freq_Units

int32 __CFUNC DAQmxGetAIFreqUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIFreqUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIFreqUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIFreqUnits gets the Analog Input >> Frequency >> Units property.

DAQmxSetAIFreqUnits sets the Analog Input >> Frequency >> Units property.

DAQmxResetAIFreqUnits resets the Analog Input >> Frequency >> Units property.
Get/Set/Reset AI_Freq_ThreshVoltage

int32 __CFUNC DAQmxGetAIFreqThreshVoltage(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIFreqThreshVoltage(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIFreqThreshVoltage(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIFreqThreshVoltage gets the Analog Input >> Frequency >> Voltage >> Threshold Level property.

DAQmxSetAIFreqThreshVoltage sets the Analog Input >> Frequency >> Voltage >> Threshold Level property.

DAQmxResetAIFreqThreshVoltage resets the Analog Input >> Frequency >> Voltage >> Threshold Level property.
Get/Set/Reset AI_Freq_Hyst

int32 __CFUNC DAQmxGetAIFreqHyst(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIFreqHyst(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIFreqHyst(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIFreqHyst gets the Analog Input >> Frequency >> Voltage >> Hysteresis property.

DAQmxSetAIFreqHyst sets the Analog Input >> Frequency >> Voltage >> Hysteresis property.

DAQmxResetAIFreqHyst resets the Analog Input >> Frequency >> Voltage >> Hysteresis property.
Get/Set/Reset AI_LVDT_Units

int32 __CFUNC DAQmxGetAILVDTUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAILVDTUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAILVDTUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAILVDTUnits gets the Analog Input >> Position >> LVDT >> Units property.
DAQmxSetAILVDTUnits sets the Analog Input >> Position >> LVDT >> Units property.
DAQmxResetAILVDTUnits resets the Analog Input >> Position >> LVDT >> Units property.
Get/Set/Reset AI_LVDT_Sensitivity

int32 __CFUNC DAQmxGetAILVDTSSensitivity(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAILVDTSSensitivity(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAILVDTSSensitivity(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAILVDTDSensitivity gets the Analogue Input >> Position >> LVDT >> Sensitivity property.

DAQmxSetAILVDTDSensitivity sets the Analogue Input >> Position >> LVDT >> Sensitivity property.

DAQmxResetAILVDTDSensitivity resets the Analogue Input >> Position >> LVDT >> Sensitivity property.
Get/Set/Reset AI_LVDT_SensitivityUnits

int32 __CFUNC DAQmxGetAILVDTSensitivityUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAILVDTSensitivityUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAILVDTSensitivityUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAILVDTSSensitivityUnits gets the Analog Input >> Position >> LVDT >> Sensitivity Units property.

DAQmxSetAILVDTSSensitivityUnits sets the Analog Input >> Position >> LVDT >> Sensitivity Units property.

DAQmxResetAILVDTSSensitivityUnits resets the Analog Input >> Position >> LVDT >> Sensitivity Units property.
Get/Set/Reset AI_RVDT_Units

int32 __CFUNC DAQmxGetAIRVDTUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIRVDTUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIRVDTUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIRVDTUnits gets the Analog Input >> Position >> RVDT >> Units property.

DAQmxSetAIRVDTUnits sets the Analog Input >> Position >> RVDT >> Units property.

DAQmxResetAIRVDTUnits resets the Analog Input >> Position >> RVDT >> Units property.
Get/Set/Reset AI_RVDT_Sensitivity

int32 __CFUNC DAQmxGetAIRVDTSensitivity(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIRVDTSensitivity(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIRVDTSensitivity(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIRVDTSSensitivity gets the Analog Input >> Position >> RVDT >> Sensitivity property.

DAQmxSetAIRVDTSSensitivity sets the Analog Input >> Position >> RVDT >> Sensitivity property.

DAQmxResetAIRVDTSSensitivity resets the Analog Input >> Position >> RVDT >> Sensitivity property.
Get/Set/Reset AI_RVDT_SensitivityUnits

int32 __CFUNC DAQmxGetAIRVDTSSensitivityUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIRVDTSSensitivityUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIRVDTSSensitivityUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIRVDTSSensitivityUnits gets the Analog Input >> Position >> RVDT >> Sensitivity Units property.

DAQmxSetAIRVDTSSensitivityUnits sets the Analog Input >> Position >> RVDT >> Sensitivity Units property.

DAQmxResetAIRVDTSSensitivityUnits resets the Analog Input >> Position >> RVDT >> Sensitivity Units property.
Get/Set/Reset
AI_SoundPressure_MaxSoundPressureLvl

int32 __CFUNC
    DAQmxGetAISoundPressureMaxSoundPressureLvl(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
    DAQmxSetAISoundPressureMaxSoundPressureLvl(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetAISoundPressureMaxSoundPressureLvl(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAISoundPressureMaxSoundPressureLvl gets the **Analog Input >> Sound Pressure >> Maximum Sound Pressure Level** property.

DAQmxSetAISoundPressureMaxSoundPressureLvl sets the **Analog Input >> Sound Pressure >> Maximum Sound Pressure Level** property.

DAQmxResetAISoundPressureMaxSoundPressureLvl resets the **Analog Input >> Sound Pressure >> Maximum Sound Pressure Level** property.
Get/Set/Reset AI_SoundPressure_Units

int32 __CFUNC DAQmxGetAISoundPressureUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAISoundPressureUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAISoundPressureUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAISoundPressureUnits gets the Analog Input >> Sound Pressure >> Units property.

DAQmxSetAISoundPressureUnits sets the Analog Input >> Sound Pressure >> Units property.

DAQmxResetAISoundPressureUnits resets the Analog Input >> Sound Pressure >> Units property.
Get/Set/Reset AI_SoundPressure_dBRef

int32 __CFUNC DAQmxGetAISoundPressuredBRef(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAISoundPressuredBRef(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAISoundPressuredBRef(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAISoundPressuredBRef gets the Analog Input >> Sound Pressure >> dB Reference property.

DAQmxSetAISoundPressuredBRef sets the Analog Input >> Sound Pressure >> dB Reference property.

DAQmxResetAISoundPressuredBRef resets the Analog Input >> Sound Pressure >> dB Reference property.
Get/Set/Reset AI_Microphone_Sensitivity

```c
int32 __CFUNC DAQmxGetAIMicrophoneSensitivity(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIMicrophoneSensitivity(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIMicrophoneSensitivity(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAIMicrophoneSensitivity gets the Analog Input >> Sound Pressure >> Microphone >> Sensitivity property.

DAQmxSetAIMicrophoneSensitivity sets the Analog Input >> Sound Pressure >> Microphone >> Sensitivity property.

DAQmxResetAIMicrophoneSensitivity resets the Analog Input >> Sound Pressure >> Microphone >> Sensitivity property.
Get/Set/Reset AI_Accel_Units

int32 __CFUNC DAQmxGetAIAccelUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIAccelUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIAccelUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIAccelUnits gets the Analog Input >> Acceleration >> Units property.
DAQmxSetAIAccelUnits sets the Analog Input >> Acceleration >> Units property.
DAQmxResetAIAccelUnits resets the Analog Input >> Acceleration >> Units property.
Get/Set/Reset AI_Accel_dBRef

int32 __CFUNC DAQmxGetAI AcceldBRef(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAI AcceldBRef(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAI AcceldBRef(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIAcceldBRef gets the [Analog Input >> Acceleration >> dB Reference](https://example.com) property.

DAQmxSetAIAcceldBRef sets the [Analog Input >> Acceleration >> dB Reference](https://example.com) property.

DAQmxResetAIAcceldBRef resets the [Analog Input >> Acceleration >> dB Reference](https://example.com) property.
Get/Set/Reset AI_Accel_Sensitivity

int32 __CFUNC DAQmxGetAIAccelSensitivity(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIAccelSensitivity(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIAccelSensitivity(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIAccelSensitivity gets the Analog Input >> Acceleration >> Accelerometer >> Sensitivity property.

DAQmxSetAIAccelSensitivity sets the Analog Input >> Acceleration >> Accelerometer >> Sensitivity property.

DAQmxResetAIAccelSensitivity resets the Analog Input >> Acceleration >> Accelerometer >> Sensitivity property.
Get/Set/Reset AI_Accel_SensitivityUnits

int32 __CFUNC DAQmxGetAIAccelSensitivityUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIAccelSensitivityUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIAccelSensitivityUnits(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIAccelSensitivityUnits gets the Analog Input >> Acceleration >> Accelerometer >> Sensitivity Units property.

DAQmxSetAIAccelSensitivityUnits sets the Analog Input >> Acceleration >> Accelerometer >> Sensitivity Units property.

DAQmxResetAIAccelSensitivityUnits resets the Analog Input >> Acceleration >> Accelerometer >> Sensitivity Units property.
Get/Set/Reset AI_Is_TEDS

int32 __CFUNC DAQmxGetAIIsTEDS(TaskHandle taskHandle, const char channel[], bool32 *data);
Purpose

DAQmxGetAIIstTEDS gets the Analog Input >> TEDS >> Is TEDS property.
Get/Set/Reset AI_TEDS_Units

int32 __CFUNC DAQmxGetAITEDSUunits(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetAITEDSUnits gets the Analog Input >> TEDS >> Units property.
**Get/Set/Reset AI_Coupling**

```
int32 __CFUNC DAQmxGetAICoupling(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAICoupling(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAICoupling(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAICoupling gets the property.

DAQmxSetAICoupling sets the property.

DAQmxResetAICoupling resets the property.
Get/Set/Reset AI_Impedance

int32 __CFUNC DAQmxGetAIImpedance(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIImpedance(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIImpedance(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIImpedance gets the property.
DAQmxSetAIImpedance sets the property.
DAQmxResetAIImpedance resets the property.
**Get/Set/Reset AI_TermCfg**

```c
int32 __CFUNC DAQmxGetAITermCfg(TaskHandle taskHandle, const char *channel[], int32 *data);

int32 __CFUNC DAQmxSetAITermCfg(TaskHandle taskHandle, const char *channel[], int32 data);

int32 __CFUNC DAQmxResetAITermCfg(TaskHandle taskHandle, const char *channel[]);
```
Purpose

DAQmxGetAITermCfg gets the property.

DAQmxSetAITermCfg sets the property.

DAQmxResetAITermCfg resets the property.
Get/Set/Reset AI_InputSrc

int32 __CFUNC DAQmxGetAIInputSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAIInputSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAIInputSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIInputSrc gets the Analog Input >> General Properties >> Input Configuration >> Input Source property.

DAQmxSetAIInputSrc sets the Analog Input >> General Properties >> Input Configuration >> Input Source property.

DAQmxResetAIInputSrc resets the Analog Input >> General Properties >> Input Configuration >> Input Source property.
Get/Set/Reset AI_ResistanceCfg

int32 __CFUNC DAQmxGetAIResistanceCfg(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIResistanceCfg(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIResistanceCfg(TaskHandle taskHandle, const char channel[]);
Purpose


DAQmxSetAIResistanceCfg sets the [Analog Input >> General Properties >> Signal Conditioning >> Resistance Configuration] property.

DAQmxResetAIResistanceCfg resets the [Analog Input >> General Properties >> Signal Conditioning >> Resistance Configuration] property.
Get/Set/Reset AI_LeadWireResistance

```c
int32 __CFUNC DAQmxGetAILeadWireResistance(TaskHandle taskHandle,
                                           const char channel[],
                                           float64 *data);

int32 __CFUNC DAQmxSetAILeadWireResistance(TaskHandle taskHandle,
                                            const char channel[],
                                            float64 data);

int32 __CFUNC DAQmxResetAILeadWireResistance(TaskHandle taskHandle,
                                              const char channel[]);
```
Purpose

DAQmxGetAILeadWireResistance gets the Lead Wire Resistance property.

DAQmxSetAILeadWireResistance sets the Lead Wire Resistance property.

DAQmxResetAILeadWireResistance resets the Lead Wire Resistance property.
Get/Set/Reset AI_Bridge_Cfg

int32 __CFUNC DAQmxGetAIBridgeCfg(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIBridgeCfg(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIBridgeCfg(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIBridgeCfg gets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Configuration property.

DAQmxSetAIBridgeCfg sets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Configuration property.

DAQmxResetAIBridgeCfg resets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Configuration property.
Get/Set/Reset AI_Bridge NomResistance

int32 __CFUNC DAQmxGetAIBridgeNomResistance(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIBridgeNomResistance(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIBridgeNomResistance(TaskHandle taskHandle, const char channel[]);
**Purpose**


DAQmxSetAIBridgeNomResistance sets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Nominal Resistance property.

DAQmxResetAIBridgeNomResistance resets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Nominal Resistance property.
Get/Set/Reset AI_Bridge_InitialVoltage

int32 __CFUNC DAQmxGetAIBridgeInitialVoltage(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIBridgeInitialVoltage(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIBridgeInitialVoltage(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIBridgeInitialVoltage gets the property.

DAQmxSetAIBridgeInitialVoltage sets the property.

DAQmxResetAIBridgeInitialVoltage resets the property.
Get/Set/Reset AI_Bridge_ShuntCal_Enable

```c
int32 __CFUNC DAQmxGetAIBridgeShuntCalEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIBridgeShuntCalEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIBridgeShuntCalEnable(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAIBridgeShuntCalEnable gets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Shunt Cal Enable property.

DAQmxSetAIBridgeShuntCalEnable sets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Shunt Cal Enable property.

DAQmxResetAIBridgeShuntCalEnable resets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Shunt Cal Enable property.
Get/Set/Reset  AI_Bridge_ShuntCal_Select

int32 __CFUNC DAQmxGetAIBridgeShuntCalSelect(TaskHandle taskHandle, 
const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIBridgeShuntCalSelect(TaskHandle taskHandle, 
const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIBridgeShuntCalSelect(TaskHandle 
taskHandle, const char channel[]);
Purpose

DAQmxGetAIBridgeShuntCalSelect gets the property.

DAQmxSetAIBridgeShuntCalSelect sets the property.

DAQmxResetAIBridgeShuntCalSelect resets the property.
Get/Set/Reset AI_Bridge_ShuntCal_GainAdjust

int32 __CFUNC DAQmxGetAIBridgeShuntCalGainAdjust(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIBridgeShuntCalGainAdjust(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIBridgeShuntCalGainAdjust(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIBridgeShuntCalGainAdjust gets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Gain Adjustment property.

DAQmxSetAIBridgeShuntCalGainAdjust sets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Gain Adjustment property.

DAQmxResetAIBridgeShuntCalGainAdjust resets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Shunt Cal >> Gain Adjustment property.
Get/Set/Reset AI_Bridge_Balance_CoarsePot

int32 __CFUNC DAQmxGetAIBridgeBalanceCoarsePot(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIBridgeBalanceCoarsePot(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIBridgeBalanceCoarsePot(TaskHandle taskHandle, const char channel[]);
**Purpose**


DAQmxSetAIBridgeBalanceCoarsePot sets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Coarse Potentiometer property.

DAQmxResetAIBridgeBalanceCoarsePot resets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Coarse Potentiometer property.
Get/Set/Reset AI_Bridge_Balance_FinePot

int32 __CFUNC DAQmxGetAIBridgeBalanceFinePot(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIBridgeBalanceFinePot(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIBridgeBalanceFinePot(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIBridgeBalanceFinePot gets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Fine Potentiometer property.

DAQmxSetAIBridgeBalanceFinePot sets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Fine Potentiometer property.

DAQmxResetAIBridgeBalanceFinePot resets the Analog Input >> General Properties >> Signal Conditioning >> Bridge >> Balance >> Fine Potentiometer property.
Get/Set/Reset AI_CurrentShunt_Loc

int32 __CFUNC DAQmxGetAICurrentShuntLoc(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAICurrentShuntLoc(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAICurrentShuntLoc(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAICurrentShuntLoc gets the [Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Location](#) property.

DAQmxSetAICurrentShuntLoc sets the [Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Location](#) property.

DAQmxResetAICurrentShuntLoc resets the [Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Location](#) property.
Get/Set/Reset AI_CurrentShunt_Resistance

```c
int32 __CFUNC DAQmxGetAICurrentShuntResistance(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAICurrentShuntResistance(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAICurrentShuntResistance(TaskHandle taskHandle, const char channel[]);
```
**Purpose**


DAQmxSetAICurrentShuntResistance sets the Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Value property.

DAQmxResetAICurrentShuntResistance resets the Analog Input >> General Properties >> Signal Conditioning >> Current Shunt Resistor >> Value property.
Get/Set/Reset AI_Excit_Src

int32 __CFUNC DAQmxGetAIExcitSrc(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIExcitSrc(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIExcitSrc(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIExcitSrc gets the property.

DAQmxSetAIExcitSrc sets the property.

DAQmxResetAIExcitSrc resets the property.
Get/Set/Reset AI_Excit_Val

int32 __CFUNC DAQmxGetAIExcitVal(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIExcitVal(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIExcitVal(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIExcitVal gets the property.

DAQmxSetAIExcitVal sets the property.

DAQmxResetAIExcitVal resets the property.
Get/Set/Reset AI_Excit_UseForScaling

int32 __CFUNC DAQmxGetAIExcitUseForScaling(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIExcitUseForScaling(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIExcitUseForScaling(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIExcitUseForScaling gets the property.

DAQmxSetAIExcitUseForScaling sets the property.

DAQmxResetAIExcitUseForScaling resets the property.
Get/Set/Reset AI_Excit_UseMultiplexed

int32 __CFUNC DAQmxGetAIExcitUseMultiplexed(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIExcitUseMultiplexed(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIExcitUseMultiplexed(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIExcitUseMultiplexed gets the Analog Input > General Properties > Signal Conditioning > Excitation > Advanced > Use Multiplexed Excitation property.

DAQmxSetAIExcitUseMultiplexed sets the Analog Input > General Properties > Signal Conditioning > Excitation > Advanced > Use Multiplexed Excitation property.

DAQmxResetAIExcitUseMultiplexed resets the Analog Input > General Properties > Signal Conditioning > Excitation > Advanced > Use Multiplexed Excitation property.
Get/Set/Reset AI_Excit_ActualVal

int32 __CFUNC DAQmxGetAIExcitActualVal(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIExcitActualVal(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIExcitActualVal(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIExcitActualVal gets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Actual Excitation Value property.

DAQmxSetAIExcitActualVal sets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Actual Excitation Value property.

DAQmxResetAIExcitActualVal resets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Actual Excitation Value property.
Get/Set/Reset AI_Excit_DCorAC

int32 __CFUNC DAQmxGetAIExcitDCorAC(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIExcitDCorAC(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIExcitDCorAC(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIExcitDCorAC gets the [Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> DC or AC] property.

DAQmxSetAIExcitDCorAC sets the [Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> DC or AC] property.

DAQmxResetAIExcitDCorAC resets the [Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> DC or AC] property.
Get/Set/Reset AI_Excit_VoltageOrCurrent

int32 __CFUNC DAQmxGetAIExcitVoltageOrCurrent(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIExcitVoltageOrCurrent(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIExcitVoltageOrCurrent(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIExcitVoltageOrCurrent gets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Voltage or Current property.

DAQmxSetAIExcitVoltageOrCurrent sets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Voltage or Current property.

DAQmxResetAIExcitVoltageOrCurrent resets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> Advanced >> Voltage or Current property.
Get/Set/Reset AI_ACExcit_Freq

int32 __CFUNC DAQmxGetAIACExcitFreq(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIACExcitFreq(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIACExcitFreq(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIACExcitFreq gets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> AC Excitation >> Frequency property.

DAQmxSetAIACExcitFreq sets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> AC Excitation >> Frequency property.

DAQmxResetAIACExcitFreq resets the Analog Input >> General Properties >> Signal Conditioning >> Excitation >> AC Excitation >> Frequency property.
Get/Set/Reset AI_ACExcit_SyncEnable

int32 __CFUNC DAQmxGetAIACExcitSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIACExcitSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIACExcitSyncEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIACExcitSyncEnable gets the Analog Input >> General Properties >> Signal Conditioning >> AC Excitation >> Synchronization Enable property.

DAQmxSetAIACExcitSyncEnable sets the Analog Input >> General Properties >> Signal Conditioning >> AC Excitation >> Synchronization Enable property.

DAQmxResetAIACExcitSyncEnable resets the Analog Input >> General Properties >> Signal Conditioning >> AC Excitation >> Synchronization Enable property.
Get/Set/Reset AI_ACExcit_WireMode

int32 __CFUNC DAQmxGetAIACExcitWireMode(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIACExcitWireMode(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIACExcitWireMode(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIACExcitWireMode gets the property.

DAQmxSetAIACExcitWireMode sets the property.

DAQmxResetAIACExcitWireMode resets the property.
Get/Set/Reset AI_Atten

int32 __CFUNC DAQmxGetAIAtten(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIAtten(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIAtten(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIAtten gets the {Attenuation Value} property.

DAQmxSetAIAtten sets the {Attenuation Value} property.

DAQmxResetAIAtten resets the {Attenuation Value} property.
Get/Set/Reset AI_Lowpass_Enable

int32 __CFUNC DAQmxGetAILowpassEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAILowpassEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAILowpassEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAILowpassEnable gets the **Enable** property.

DAQmxSetAILowpassEnable sets the **Enable** property.

DAQmxResetAILowpassEnable resets the **Enable** property.
**Get/Set/Reset AI_Lowpass_CutoffFreq**

```c
int32 __CFUNC DAQmxGetAILowpassCutoffFreq(TaskHandle taskHandle,
                                           const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAILowpassCutoffFreq(TaskHandle taskHandle,
                                           const char channel[], float64 data);

int32 __CFUNC DAQmxResetAILowpassCutoffFreq(TaskHandle taskHandle,
                                             const char channel[]);
```
Purpose

DAQmxGetAILowpassCutoffFreq gets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Cutoff Frequency property.

DAQmxSetAILowpassCutoffFreq sets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Cutoff Frequency property.

DAQmxResetAILowpassCutoffFreq resets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Cutoff Frequency property.
Get/Set/Reset AI_Lowpass_SwitchCap_ClkSrc

int32 __CFUNC DAQmxGetAILowpassSwitchCapClkSrc(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAILowpassSwitchCapClkSrc(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAILowpassSwitchCapClkSrc(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAILowpassSwitchCapClkSrc gets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> Clock Source property.

DAQmxSetAILowpassSwitchCapClkSrc sets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> Clock Source property.

DAQmxResetAILowpassSwitchCapClkSrc resets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> Clock Source property.
Get/Set/Reset AI_Lowpass_SwitchCap_ExtClkFreq

int32 __CFUNC DAQmxGetAILowpassSwitchCapExtClkFreq(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAILowpassSwitchCapExtClkFreq(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAILowpassSwitchCapExtClkFreq(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAILowpassSwitchCapExtClkFreq gets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Frequency property.

DAQmxSetAILowpassSwitchCapExtClkFreq sets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Frequency property.

DAQmxResetAILowpassSwitchCapExtClkFreq resets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Frequency property.
Get/Set/Reset AI_Lowpass_SwitchCap_ExtClkDiv

int32 __CFUNC DAQmxGetAILowpassSwitchCapExtClkDiv(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetAILowpassSwitchCapExtClkDiv(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetAILowpassSwitchCapExtClkDiv(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAILowpassSwitchCapExtClkDiv gets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Divisor property.

DAQmxSetAILowpassSwitchCapExtClkDiv sets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Divisor property.

DAQmxResetAILowpassSwitchCapExtClkDiv resets the Analog Input >> General Properties >> Filter >> Analog Lowpass >> Advanced >> Switched Capacitor >> External Clock Divisor property.
Get/Set/Reset AI_Lowpass_SwitchCap_OutClkDiv

int32 __CFUNC DAQmxGetAILowpassSwitchCapOutClkDiv(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetAILowpassSwitchCapOutClkDiv(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetAILowpassSwitchCapOutClkDiv(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAILowpassSwitchCapOutClkDiv gets the Analog Input >> General Properties >> Filter Analog Lowpass >> Advanced >> Switched Capacitor >> Output Clock Divisor property.

DAQmxSetAILowpassSwitchCapOutClkDiv sets the Analog Input >> General Properties >> Filter Analog Lowpass >> Advanced >> Switched Capacitor >> Output Clock Divisor property.

DAQmxResetAILowpassSwitchCapOutClkDiv resets the Analog Input >> General Properties >> Filter Analog Lowpass >> Advanced >> Switched Capacitor >> Output Clock Divisor property.
Get/Set/Reset AI_ResolutionUnits

int32 __CFUNC DAQmxGetAIResolutionUnits(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetAIResolutionUnits gets the Analog Input >> General Properties >> Digitizer/ADC >> Resolution Units property.
Get/Set/Reset AI_Resolution

int32 __CFUNC DAQmxGetAIResolution(TaskHandle taskHandle, const char channel[], float64 *data);
Purpose

DAQmxGetAIResolution gets the Analog Input >> General Properties >> Digitizer/ADC >> Resolution Value property.
Get/Set/Reset AI_RawSampSize

int32 __CFUNC DAQmxGetAIRawSampSize(TaskHandle taskHandle, const char channel[], uInt32 *data);
Purpose

DAQmxGetAIRawSampSize gets the Analog Input >> General Properties >> Digitizer/ADC >> Raw Sample Size property.
Get/Set/Reset AI_RawSampJustification

int32 __CFUNC DAQmxGetAIrawSampJustification(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetAIRawSampJustification gets the Analog Input >> General Properties >> Digitizer/ADC >> Raw Sample Justification property.
Get/Set/Reset AI_ADCTimingMode

int32 __CFUNC DAQmxGetAIADCTimingMode(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIADCTimingMode(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIADCTimingMode(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIADCTimingMode gets the Analog Input >> General Properties >> Digitizer/ADC >> Timing Mode property.

DAQmxSetAIADCTimingMode sets the Analog Input >> General Properties >> Digitizer/ADC >> Timing Mode property.

DAQmxResetAIADCTimingMode resets the Analog Input >> General Properties >> Digitizer/ADC >> Timing Mode property.
Get/Set/Reset AI_Dither_Enable

int32 __CFUNC DAQmxGetAIDitherEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIDitherEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIDitherEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIDitherEnable gets the property.

DAQmxSetAIDitherEnable sets the property.

DAQmxResetAIDitherEnable resets the property.
Get/Set/Reset AI_ChanCal_HasValidCalInfo

int32 __CFUNC DAQmxGetAIChanCalHasValidCalInfo(TaskHandle taskHandle, const char channel[], bool32 *data);
Purpose

DAQmxGetAIChanCalHasValidCalInfo gets the Analog Input >> General Properties >> Channel Calibration >> Has Valid Calibration Information property.
Get/Set/Reset AI_ChanCal_EnableCal

int32 __CFUNC DAQmxGetAIChanCalEnableCal(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIChanCalEnableCal(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIChanCalEnableCal(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalEnableCal gets the Analog Input >> General Properties >> Channel Calibration >> Enable Calibration property.

DAQmxSetAIChanCalEnableCal sets the Analog Input >> General Properties >> Channel Calibration >> Enable Calibration property.

DAQmxResetAIChanCalEnableCal resets the Analog Input >> General Properties >> Channel Calibration >> Enable Calibration property.
Get/Set/Reset AI_ChanCal_ApplyCalIfExp

int32 __CFUNC DAQmxGetAIChanCalApplyCalIfExp(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIChanCalApplyCalIfExp(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIChanCalApplyCalIfExp(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalApplyCalIfExp gets the property.

DAQmxSetAIChanCalApplyCalIfExp sets the property.

DAQmxResetAIChanCalApplyCalIfExp resets the property.
Get/Set/Reset AI_ChanCal_ScaleType

int32 __CFUNC DAQmxGetAIChanCalScaleType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIChanCalScaleType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIChanCalScaleType(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalScaleType gets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Scale Type property.

DAQmxSetAIChanCalScaleType sets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Scale Type property.

DAQmxResetAIChanCalScaleType resets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Scale Type property.
Get/Set/Reset AI_ChanCal_Table_PreScaledVals

int32 __CFUNC DAQmxGetAIChanCalTablePreScaledVals(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetAIChanCalTablePreScaledVals(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxResetAIChanCalTablePreScaledVals(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIChanCalTablePreScaledVals gets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Pre-Scaled Values property.

DAQmxSetAIChanCalTablePreScaledVals sets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Pre-Scaled Values property.

DAQmxResetAIChanCalTablePreScaledVals resets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Pre-Scaled Values property.
Get/Set/Reset AI_ChanCal_Table_ScaledVals

int32 __CFUNC DAQmxGetAIChanCalTableScaledVals(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetAIChanCalTableScaledVals(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxResetAIChanCalTableScaledVals(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalTableScaledVals gets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Scaled Values property.

DAQmxSetAIChanCalTableScaledVals sets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Scaled Values property.

DAQmxResetAIChanCalTableScaledVals resets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Table >> Scaled Values property.
Get/Set/Reset AI_ChanCal_Poly_ForwardCoeff

int32 __CFUNC DAQmxGetAIChanCalPolyForwardCoeff(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetAIChanCalPolyForwardCoeff(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxResetAIChanCalPolyForwardCoeff(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAIChanCalPolyForwardCoeff gets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Forward Coefficients property.

DAQmxSetAIChanCalPolyForwardCoeff sets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Forward Coefficients property.

DAQmxResetAIChanCalPolyForwardCoeff resets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Forward Coefficients property.
Get/Set/Reset AI_ChanCal_Poly.ReverseCoeff

int32 __CFUNC DAQmxGetAIChanCalPolyReverseCoeff(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetAIChanCalPolyReverseCoeff(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxResetAIChanCalPolyReverseCoeff(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalPolyReverseCoeff gets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Reverse Coefficients property.

DAQmxSetAIChanCalPolyReverseCoeff sets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Reverse Coefficients property.

DAQmxResetAIChanCalPolyReverseCoeff resets the Analog Input >> General Properties >> Channel Calibration >> Scaling Parameters >> Polynomial >> Reverse Coefficients property.
Get/Set/Reset AI_ChanCal_OperatorName

int32 __CFUNC DAQmxGetAIChanCalOperatorName(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAIChanCalOperatorName(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAIChanCalOperatorName(TaskHandle taskHandle, const char channel[]);
**Purpose**

**DAQmxGetAIChanCalOperatorName** gets the [Analog Input >> General Properties >> Channel Calibration](#) property.

**DAQmxSetAIChanCalOperatorName** sets the [Analog Input >> General Properties >> Channel Calibration](#) property.

**DAQmxResetAIChanCalOperatorName** resets the [Analog Input >> General Properties >> Channel Calibration](#) property.
Get/Set/Reset AI_ChanCal_Desc

int32 __CFUNC DAQmxGetAIChanCalDesc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAIChanCalDesc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAIChanCalDesc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalDesc gets the Analog Input >> General Properties >> Channel Calibration >> Description property.

DAQmxSetAIChanCalDesc sets the Analog Input >> General Properties >> Channel Calibration >> Description property.

DAQmxResetAIChanCalDesc resets the Analog Input >> General Properties >> Channel Calibration >> Description property.
Get/Set/Reset AI_ChanCal_Verif_RefVals

```c
int32 __CFUNC DAQmxGetAIChanCalVerifRefVals(TaskHandle taskHandle,
                                             const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetAIChanCalVerifRefVals(TaskHandle taskHandle,
                                             const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxResetAIChanCalVerifRefVals(TaskHandle taskHandle,
                                             const char channel[]);
```
Purpose

DAQmxGetAIChanCalVerifRefVals gets the Analog Input >> General Properties >> Channel Calibration >> Verification >> Reference Values property.

DAQmxSetAIChanCalVerifRefVals sets the Analog Input >> General Properties >> Channel Calibration >> Verification >> Reference Values property.

DAQmxResetAIChanCalVerifRefVals resets the Analog Input >> General Properties >> Channel Calibration >> Verification >> Reference Values property.
Get/Set/Reset AI_ChanCal_Verif_AcqVals

int32 __CFUNC DAQmxGetAIChanCalVerifAcqVals(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetAIChanCalVerifAcqVals(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxResetAIChanCalVerifAcqVals(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIChanCalVerifAcqVals gets the Analog Input >> General Properties >> Channel Calibration >> Verification >> Acquired Values property.

DAQmxSetAIChanCalVerifAcqVals sets the Analog Input >> General Properties >> Channel Calibration >> Verification >> Acquired Values property.

DAQmxResetAIChanCalVerifAcqVals resets the Analog Input >> General Properties >> Channel Calibration >> Verification >> Acquired Values property.
**Get/Set/Reset AI_Rng_High**

```c
int32 __CFUNC DAQmxGetAIRngHigh(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIRngHigh(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIRngHigh(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAIRngHigh gets the Analog Input >> General Properties >> Advanced >> Range >> High property.

DAQmxSetAIRngHigh sets the Analog Input >> General Properties >> Advanced >> Range >> High property.

DAQmxResetAIRngHigh resets the Analog Input >> General Properties >> Advanced >> Range >> High property.
**Get/Set/Reset AI_Rng_Low**

```c
int32 __CFUNC DAQmxGetAIRngLow(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIRngLow(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIRngLow(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetAIRngLow gets the [Analog Input >> General Properties >> Advanced >> Range >> Low](#) property.

DAQmxSetAIRngLow sets the [Analog Input >> General Properties >> Advanced >> Range >> Low](#) property.

DAQmxResetAIRngLow resets the [Analog Input >> General Properties >> Advanced >> Range >> Low](#) property.
Get/Set/Reset AI_Gain

int32 __CFUNC DAQmxGetAIGain(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAIGain(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAIGain(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIGain gets the Analog Input >> General Properties >> Advanced >> Gain >> Gain Value property.

DAQmxSetAIGain sets the Analog Input >> General Properties >> Advanced >> Gain >> Gain Value property.

DAQmxResetAIGain resets the Analog Input >> General Properties >> Advanced >> Gain >> Gain Value property.
Get/Set/Reset AI_SampAndHold_Enable

int32 __CFUNC DAQmxGetAISampAndHoldEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAISampAndHoldEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAISampAndHoldEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAISampAndHoldEnable gets the **Analog Input >> General Properties >> Advanced >> Sample and Hold Enable** property.

DAQmxSetAISampAndHoldEnable sets the **Analog Input >> General Properties >> Advanced >> Sample and Hold Enable** property.

DAQmxResetAISampAndHoldEnable resets the **Analog Input >> General Properties >> Advanced >> Sample and Hold Enable** property.
Get/Set/Reset AI_AutoZeroMode

int32 __CFUNC DAQmxGetAIAutoZeroMode(TaskHandle taskHandle, const char *channel[], int32 *data);

int32 __CFUNC DAQmxSetAIAutoZeroMode(TaskHandle taskHandle, const char *channel[], int32 data);

int32 __CFUNC DAQmxResetAIAutoZeroMode(TaskHandle taskHandle, const char *channel[]);
Purpose

DAQmxGetAIAutoZeroMode gets the Analog Input >> General Properties >> Advanced >> High Accuracy Settings >> Auto Zero Mode property.

DAQmxSetAIAutoZeroMode sets the Analog Input >> General Properties >> Advanced >> High Accuracy Settings >> Auto Zero Mode property.

DAQmxResetAIAutoZeroMode resets the Analog Input >> General Properties >> Advanced >> High Accuracy Settings >> Auto Zero Mode property.
Get/Set/Reset AI_DataXferMech

int32 __CFUNC DAQmxGetAIDataXferMech(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIDataXferMech(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIDataXferMech(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIDataXferMech gets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxSetAIDataXferMech sets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxResetAIDataXferMech resets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.
Get/Set/Reset AI_DataXferReqCond

int32 __CFUNC DAQmxGetAIDataXferReqCond(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIDataXferReqCond(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIDataXferReqCond(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIDataXferReqCond gets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition property.

DAQmxSetAIDataXferReqCond sets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition property.

DAQmxResetAIDataXferReqCond resets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition property.
Get/Set/Reset AI_DataXferCustomThreshold

int32 __CFUNC DAQmxGetAIDataXferCustomThreshold(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetAIDataXferCustomThreshold(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetAIDataXferCustomThreshold(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIDataXferCustomThreshold gets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Custom Threshold property.

DAQmxSetAIDataXferCustomThreshold sets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Custom Threshold property.

DAQmxResetAIDataXferCustomThreshold resets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Custom Threshold property.
**Get/Set/Reset AI_MemMapEnable**

```c
int32 __CFUNC DAQmxGetAIMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIMemMapEnable(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAIMemMapEnable gets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.

DAQmxSetAIMemMapEnable sets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.

DAQmxResetAIMemMapEnable resets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.
**Get/Set/Reset AI_RawDataCompressionType**

```c
int32 __CFUNC DAQmxGetAIRawDataCompressionType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAIRawDataCompressionType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAIRawDataCompressionType(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAIRawDataCompressionType gets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Raw Data Compression Type property.

DAQmxSetAIRawDataCompressionType sets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Raw Data Compression Type property.

DAQmxResetAIRawDataCompressionType resets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Raw Data Compression Type property.
Get/Set/Reset
AI_LossyLSBRemoval_CompressedSampSize

int32 __CFUNC
    DAQmxGetAILossyLSBRemovalCompressedSampSize(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC
    DAQmxSetAILossyLSBRemovalCompressedSampSize(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC
    DAQmxResetAILossyLSBRemovalCompressedSampSize(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAILossyLSBRemovalCompressedSampSize gets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Lossy LSB Removal >> Compressed Sample Size property.

DAQmxSetAILossyLSBRemovalCompressedSampSize sets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Lossy LSB Removal >> Compressed Sample Size property.

DAQmxResetAILossyLSBRemovalCompressedSampSize resets the Analog Input >> General Properties >> Advanced >> Data Transfer and Memory >> Compression >> Lossy LSB Removal >> Compressed Sample Size property.
Get/Set/Reset AI_DevScalingCoeff

int32 __CFUNC DAQmxGetAIDevScalingCoeff(TaskHandle taskHandle, const char channel[], float64 *data, ulnt32 arraySizeInSamples);
Purpose

DAQmxGetAIDevScalingCoeff gets the Analog Input >> General Properties >> Advanced >> Device Scaling Coefficients >> Device Scaling Coefficients property.
Get/Set/Reset AI_EnhancedAliasRejectionEnable

int32 __CFUNC DAQmxGetAIEnhancedAliasRejectionEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAIEnhancedAliasRejectionEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAIEnhancedAliasRejectionEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAIEnhancedAliasRejectionEnable gets the Analog Input >> General Properties >> Advanced >> Enhanced Alias Rejection Enable property.

DAQmxSetAIEnhancedAliasRejectionEnable sets the Analog Input >> General Properties >> Advanced >> Enhanced Alias Rejection Enable property.

DAQmxResetAIEnhancedAliasRejectionEnable resets the Analog Input >> General Properties >> Advanced >> Enhanced Alias Rejection Enable property.
Get/Set/Reset AO_Max

int32 __CFUNC DAQmxGetAOMax(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOMax(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOMax(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOMax gets the Analog Output >> Maximum Value property.
DAQmxSetAOMax sets the Analog Output >> Maximum Value property.
DAQmxResetAOMax resets the Analog Output >> Maximum Value property.
Get/Set/Reset AO_Min

int32 __CFUNC DAQmxGetAOMin(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOMin(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOMin(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOMin gets the Analog Output >> Minimum Value property.

DAQmxSetAOMin sets the Analog Output >> Minimum Value property.

DAQmxResetAOMin resets the Analog Output >> Minimum Value property.
Get/Set/Reset AO_CustomScaleName

int32 __CFUNC DAQmxGetAOCustomScaleName(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAOCustomScaleName(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAOCustomScaleName(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAOCustomScaleName gets the Analog Output >> Custom Scale Name property.

DAQmxSetAOCustomScaleName sets the Analog Output >> Custom Scale Name property.

DAQmxResetAOCustomScaleName resets the Analog Output >> Custom Scale Name property.
Get/Set/Reset AO_OutputType

int32 __CFUNC DAQmxGetAOOutputType(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetAOOutputType gets the Analog Output >> Output Type property.
Get/Set/Reset AO_Voltage_Units

int32 __CFUNC DAQmxGetAOVoltageUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOVoltageUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOVoltageUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOVoltageUnits gets the Analog Output >> Voltage >> Units property.
DAQmxSetAOVoltageUnits sets the Analog Output >> Voltage >> Units property.
DAQmxResetAOVoltageUnits resets the Analog Output >> Voltage >> Units property.
Get/Set/Reset AO_Voltage_CurrentLimit

int32 __CFUNC DAQmxGetAOVoltageCurrentLimit(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOVoltageCurrentLimit(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOVoltageCurrentLimit(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOVoltageCurrentLimit gets the Analog Output >> Voltage >> Current Limit property.

DAQmxSetAOVoltageCurrentLimit sets the Analog Output >> Voltage >> Current Limit property.

DAQmxResetAOVoltageCurrentLimit resets the Analog Output >> Voltage >> Current Limit property.
Get/Set/Reset AO_Current_Units

int32 __CFUNC DAQmxGetAOCurrentUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOCurrentUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOCurrentUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOCurrentUnits gets the Analog Output >> Current >> Units property.

DAQmxSetAOCurrentUnits sets the Analog Output >> Current >> Units property.

DAQmxResetAOCurrentUnits resets the Analog Output >> Current >> Units property.
Get/Set/Reset AO(FuncGen_Type)

int32 __CFUNC DAQmxGetAOFuncGenType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOFuncGenType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOFuncGenType(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOFuncGenType gets the Analog Output >> Function Generation >> Type property.

DAQmxSetAOFuncGenType sets the Analog Output >> Function Generation >> Type property.

DAQmxResetAOFuncGenType resets the Analog Output >> Function Generation >> Type property.
Get/Set/Reset AO_FuncGen_Freq

int32 __CFUNC DAQmxGetAOFuncGenFreq(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOFuncGenFreq(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOFuncGenFreq(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOFuncGenFreq gets the Analog Output >> Function Generation >> Frequency property.
DAQmxSetAOFuncGenFreq sets the Analog Output >> Function Generation >> Frequency property.
DAQmxResetAOFuncGenFreq resets the Analog Output >> Function Generation >> Frequency property.
Get/Set/Reset AO_FuncGen_Amplitude

int32 __CFUNC DAQmxGetAOFuncGenAmplitude(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOFuncGenAmplitude(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOFuncGenAmplitude(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOFuncGenAmplitude gets the Analog Output >> Function Generation >> Amplitude property.

DAQmxSetAOFuncGenAmplitude sets the Analog Output >> Function Generation >> Amplitude property.

DAQmxResetAOFuncGenAmplitude resets the Analog Output >> Function Generation >> Amplitude property.
Get/Set/Reset AO(FuncGen_Offset)

int32 __CFUNC DAQmxGetAOFuncGenOffset(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOFuncGenOffset(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOFuncGenOffset(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOFuncGenOffset gets the Analog Output >> Function Generation >> Offset property.
DAQmxSetAOFuncGenOffset sets the Analog Output >> Function Generation >> Offset property.
DAQmxResetAOFuncGenOffset resets the Analog Output >> Function Generation >> Offset property.
Get/Set/Reset AO_FuncGen_Square_DutyCycle

```c
int32 __CFUNC DAQmxGetAOFuncGenSquareDutyCycle(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOFuncGenSquareDutyCycle(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOFuncGenSquareDutyCycle(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAOFuncGenSquareDutyCycle gets the Analog Output >> Function Generation >> Square >> DutyCycle property.

DAQmxSetAOFuncGenSquareDutyCycle sets the Analog Output >> Function Generation >> Square >> DutyCycle property.

DAQmxResetAOFuncGenSquareDutyCycle resets the Analog Output >> Function Generation >> Square >> DutyCycle property.
Get/Set/Reset AO_FuncGen_ModulationType

int32 __CFUNC DAQmxGetAOFuncGenModulationType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOFuncGenModulationType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOFuncGenModulationType(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOFuncGenModulationType gets the Analog Output >> Function Generation >> Modulation Type property.

DAQmxSetAOFuncGenModulationType sets the Analog Output >> Function Generation >> Modulation Type property.

DAQmxResetAOFuncGenModulationType resets the Analog Output >> Function Generation >> Modulation Type property.
Get/Set/Reset AO_FuncGen_FMDeviation

int32 __CFUNC DAQmxGetAOFuncGenFMDeviation(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOFuncGenFMDeviation(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOFuncGenFMDeviation(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOFuncGenFMDeviation gets the Analog Output >> Function Generation >> Modulation >> FM Deviation property.

DAQmxSetAOFuncGenFMDeviation sets the Analog Output >> Function Generation >> Modulation >> FM Deviation property.

DAQmxResetAOFuncGenFMDeviation resets the Analog Output >> Function Generation >> Modulation >> FM Deviation property.
Get/Set/Reset AO_OutputImpedance

int32 __CFUNC DAQmxGetAOOutputImpedance(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOOutputImpedance(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOOutputImpedance(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAOOutputImpedance gets the [Analog Output >> General Properties >> Output Configuration >> Output Impedance](#) property.

DAQmxSetAOOutputImpedance sets the [Analog Output >> General Properties >> Output Configuration >> Output Impedance](#) property.

DAQmxResetAOOutputImpedance resets the [Analog Output >> General Properties >> Output Configuration >> Output Impedance](#) property.
Get/Set/Reset AO_LoadImpedance

int32 __CFUNC DAQmxGetAOLoadImpedance(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOLoadImpedance(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOLoadImpedance(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOLoadImpedance gets the Analog Output >> General Properties >> Output Configuration >> Load Impedance property.

DAQmxSetAOLoadImpedance sets the Analog Output >> General Properties >> Output Configuration >> Load Impedance property.

DAQmxResetAOLoadImpedance resets the Analog Output >> General Properties >> Output Configuration >> Load Impedance property.
Get/Set/Reset AO_IdleOutputBehavior

int32 __CFUNC DAQmxGetAOIdleOutputBehavior(TaskHandle taskHandle,
                                           const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOIdleOutputBehavior(TaskHandle taskHandle,
                                           const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOIdleOutputBehavior(TaskHandle taskHandle,
                                              const char channel[]);
Purpose

DAQmxGetAOIdleOutputBehavior gets the Analog Output >> General Properties >> Output Configuration >> Idle Output Behavior property.

DAQmxSetAOIdleOutputBehavior sets the Analog Output >> General Properties >> Output Configuration >> Idle Output Behavior property.

DAQmxResetAOIdleOutputBehavior resets the Analog Output >> General Properties >> Output Configuration >> Idle Output Behavior property.
Get/Set/Reset AO_TermCfg

int32 __CFUNC DAQmxGetAOTermCfg(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOTermCfg(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOTermCfg(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOTermCfg gets the Analog Output >> General Properties >> Output Configuration >> Terminal Configuration property.

DAQmxSetAOTermCfg sets the Analog Output >> General Properties >> Output Configuration >> Terminal Configuration property.

DAQmxResetAOTermCfg resets the Analog Output >> General Properties >> Output Configuration >> Terminal Configuration property.
Get/Set/Reset AO_ResolutionUnits

int32 __CFUNC DAQmxGetAOResolutionUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAOResolutionUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAOResolutionUnits(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAOResolutionUnits gets the [Analog Output >> General Properties >> DAC >> Resolution Units](#) property.

DAQmxSetAOResolutionUnits sets the [Analog Output >> General Properties >> DAC >> Resolution Units](#) property.

DAQmxResetAOResolutionUnits resets the [Analog Output >> General Properties >> DAC >> Resolution Units](#) property.
Get/Set/Reset AO_Resolution

int32 __CFUNC DAQmxGetAOResolution(TaskHandle taskHandle, const char *channel[], float64 *data);
Purpose

DAQmxGetAOResolution gets the Analog Output >> General Properties >> DAC >> Resolution Value property.
Get/Set/Reset AO_DAC_Rng_High

int32 __CFUNC DAQmxGetAODACRngHigh(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAODACRngHigh(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAODACRngHigh(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAODACRngHigh gets the Analog Output >> General Properties >> DAC >> Range >> High property.

DAQmxSetAODACRngHigh sets the Analog Output >> General Properties >> DAC >> Range >> High property.

DAQmxResetAODACRngHigh resets the Analog Output >> General Properties >> DAC >> Range >> High property.
Get/Set/Reset AO_DAC_Rng_Low

```c
int32 __CFUNC DAQmxGetAODACRngLow(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAODACRngLow(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAODACRngLow(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAODACRngLow gets the Analog Output >> General Properties >> DAC >> Range >> Low property.

DAQmxSetAODACRngLow sets the Analog Output >> General Properties >> DAC >> Range >> Low property.

DAQmxResetAODACRngLow resets the Analog Output >> General Properties >> DAC >> Range >> Low property.
Get/Set/Reset AO_DAC_Ref_ConnToGnd

int32 __CFUNC DAQmxGetAO_DAC_RefConnToGnd(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAO_DAC_RefConnToGnd(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAO_DAC_RefConnToGnd(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAODACRefConnToGnd gets the property.

DAQmxSetAODACRefConnToGnd sets the property.

DAQmxResetAODACRefConnToGnd resets the property.
Get/Set/Reset AO_DAC_Ref_AllowConnToGnd

int32 __CFUNC DAQmxGetAODACRefAllowConnToGnd(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAODACRefAllowConnToGnd(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAODACRefAllowConnToGnd(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAODACRefAllowConnToGnd gets the Analog Output >> General Properties >> DAC >> Reference Voltage >> Allow Connecting DAC Reference to Ground at Runtime property.

DAQmxSetAODACRefAllowConnToGnd sets the Analog Output >> General Properties >> DAC >> Reference Voltage >> Allow Connecting DAC Reference to Ground at Runtime property.

DAQmxResetAODACRefAllowConnToGnd resets the Analog Output >> General Properties >> DAC >> Reference Voltage >> Allow Connecting DAC Reference to Ground at Runtime property.
Get/Set/Reset AO_DAC_Ref_Src

int32 __CFUNC DAQmxGetAODACRefSrc(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAODACRefSrc(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAODACRefSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAODACRefSrc gets the Analog Output >> General Properties >> DAC >> Reference Voltage >> Source property.

DAQmxSetAODACRefSrc sets the Analog Output >> General Properties >> DAC >> Reference Voltage >> Source property.

DAQmxResetAODACRefSrc resets the Analog Output >> General Properties >> DAC >> Reference Voltage >> Source property.
Get/Set/Reset AO_DAC_Ref_ExtSrc

```c
int32 __CFUNC DAQmxGetAODACRefExtSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAODACRefExtSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAODACRefExtSrc(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAODACRefExtSrc gets the **External Source** property.

DAQmxSetAODACRefExtSrc sets the **External Source** property.

DAQmxResetAODACRefExtSrc resets the **External Source** property.
Get/Set/Reset AO_DAC_Ref_Val

int32 __CFUNC DAQmxGetAODACRefVal(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAODACRefVal(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAODACRefVal(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAODACRefVal gets the property.

DAQmxSetAODACRefVal sets the property.

DAQmxResetAODACRefVal resets the property.
Get/Set/Reset AO_DAC_Offset_Src

int32 __CFUNC DAQmxGetAODACOffsetSrc(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAODACOffsetSrc(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAODACOffsetSrc(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetAODACOffsetSrc gets the [Source](#) property.

DAQmxSetAODACOffsetSrc sets the [Source](#) property.

DAQmxResetAODACOffsetSrc resets the [Source](#) property.
**Get/Set/Reset AO_DAC_Offset_ExtSrc**

```c
int32 __CFUNC DAQmxGetAODACOffsetExtSrc(TaskHandle taskHandle, const char channel[], char *data, uint32 bufferSize);

int32 __CFUNC DAQmxSetAODACOffsetExtSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetAODACOffsetExtSrc(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetAODACOffsetExtSrc gets the Analog Output >> General Properties >> DAC >> Offset Voltage >> External Source property.

DAQmxSetAODACOffsetExtSrc sets the Analog Output >> General Properties >> DAC >> Offset Voltage >> External Source property.

DAQmxResetAODACOffsetExtSrc resets the Analog Output >> General Properties >> DAC >> Offset Voltage >> External Source property.
Get/Set/Reset AO_DAC_Offset_Val

int32 __CFUNC DAQmxGetAODACOffsetVal(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAODACOffsetVal(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAODACOffsetVal(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAODACOffsetVal gets the Analog Output >> General Properties >> DAC >> Offset Voltage >> Value property.

DAQmxSetAODACOffsetVal sets the Analog Output >> General Properties >> DAC >> Offset Voltage >> Value property.

DAQmxResetAODACOffsetVal resets the Analog Output >> General Properties >> DAC >> Offset Voltage >> Value property.
Get/Set/Reset AO_ReglitchEnable

int32 __CFUNC DAQmxGetAOReglitchEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAOReglitchEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAOReglitchEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOReglitchEnable gets the property.

DAQmxSetAOReglitchEnable sets the property.

DAQmxResetAOReglitchEnable resets the property.
Get/Set/Reset AO_Gain

int32 __CFUNC DAQmxGetAOGain(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetAOGain(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetAOGain(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOGain gets the Analog Output >> General Properties >> Advanced >> Gain >> Gain Value property.

DAQmxSetAOGain sets the Analog Output >> General Properties >> Advanced >> Gain >> Gain Value property.

DAQmxResetAOGain resets the Analog Output >> General Properties >> Advanced >> Gain >> Gain Value property.
Get/Set/Reset AO_UseOnlyOnBrdMem

int32 __CFUNC DAQmxGetAOUseOnlyOnBrdMem(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAOUseOnlyOnBrdMem(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAOUseOnlyOnBrdMem(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOUseOnlyOnBrdMem gets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory property.

DAQmxSetAOUseOnlyOnBrdMem sets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory property.

DAQmxResetAOUseOnlyOnBrdMem resets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory property.
Get/Set/Reset AO_DataXferMech

int32 __CFUNC DAQmxGetAODataXferMech(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAODataXferMech(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAODataXferMech(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAODataXferMech gets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxSetAODataXferMech sets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxResetAODataXferMech resets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.
Get/Set/Reset AO_DataXferReqCond

int32 __CFUNC DAQmxGetAODataXferReqCond(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetAODataXferReqCond(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetAODataXferReqCond(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAODataXferReqCond gets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition property.

DAQmxSetAODataXferReqCond sets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition property.

DAQmxResetAODataXferReqCond resets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition property.
Get/Set/Reset AO_MemMapEnable

int32 __CFUNC DAQmxGetAOMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAOMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAOMemMapEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetAOMemMapEnable gets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.

DAQmxSetAOMemMapEnable sets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.

DAQmxResetAOMemMapEnable resets the Analog Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.
Get/Set/Reset AO_DevScalingCoeff

int32 __CFUNC DAQmxGetAODevScalingCoeff(TaskHandle taskHandle, const char channel[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetAODevScalingCoeff gets the Analog Output >> General Properties >> Advanced >> Device Scaling Coefficients >> Device Scaling Coefficients property.
Get/Set/Reset AO_EnhancedImageRejectionEnable

int32 __CFUNC DAQmxGetAOEnhancedImageRejectionEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetAOEnhancedImageRejectionEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetAOEnhancedImageRejectionEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**


DAQmxSetAOEnhancedImageRejectionEnable sets the `Analog Output >> General Properties >> Advanced >> Enhanced Image Rejection Enable` property.

DAQmxResetAOEnhancedImageRejectionEnable resets the `Analog Output >> General Properties >> Advanced >> Enhanced Image Rejection Enable` property.
Get/Set/Reset DI_InvertLines

int32 __CFUNC DAQmxGetDIInvertLines(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDIInvertLines(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDIInvertLines(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDIInvertLines gets the Digital Input >> Invert Lines property.
DAQmxSetDIInvertLines sets the Digital Input >> Invert Lines property.
DAQmxResetDIInvertLines resets the Digital Input >> Invert Lines property.
Get/Set/Reset DI_NumLines

int32 __CFUNC DAQmxGetDINumLines(TaskHandle taskHandle, const char channel[], uInt32 *data);
**Purpose**

DAQmxGetDINumLines gets the [Digital Input >> Number of Lines](https://www.ni.com) property.
Get/Set/Reset DI_DigFltr_Enable

int32 __CFUNC DAQmxGetDIDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDIDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDIDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDIDigFltrEnable gets the Digital Input >> Digital Filter >> Enable property.

DAQmxSetDIDigFltrEnable sets the Digital Input >> Digital Filter >> Enable property.

DAQmxResetDIDigFltrEnable resets the Digital Input >> Digital Filter >> Enable property.
Get/Set/Reset DI_DigFltr_MinPulseWidth

int32 __CFUNC DAQmxGetDIDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetDIDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetDIDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDIDigFltrMinPulseWidth gets the Digital Input >> Digital Filter >> Minimum Pulse Width property.

DAQmxSetDIDigFltrMinPulseWidth sets the Digital Input >> Digital Filter >> Minimum Pulse Width property.

DAQmxResetDIDigFltrMinPulseWidth resets the Digital Input >> Digital Filter >> Minimum Pulse Width property.
Get/Set/Reset DI_Tristate

int32 __CFUNC DAQmxGetDITristate(TaskHandle taskHandle, const char *channel[], bool32 *data);

int32 __CFUNC DAQmxSetDITristate(TaskHandle taskHandle, const char *channel[], bool32 data);

int32 __CFUNC DAQmxResetDITristate(TaskHandle taskHandle, const char *channel[]);
**Purpose**

DAQmxGetDITristate gets the Digital Input >> Tristate property.

DAQmxSetDITristate sets the Digital Input >> Tristate property.

DAQmxResetDITristate resets the Digital Input >> Tristate property.
Get/Set/Reset DI.LogicFamily

int32 __CFUNC DAQmxGetDILogicFamily(TaskHandle taskHandle, const char* channel[], int32 *data);

int32 __CFUNC DAQmxSetDILogicFamily(TaskHandle taskHandle, const char* channel[], int32 data);

int32 __CFUNC DAQmxResetDILogicFamily(TaskHandle taskHandle, const char* channel[]);
Purpose

DAQmxGetDILogicFamily gets the Digital Input >> Logic Family property.
DAQmxSetDILogicFamily sets the Digital Input >> Logic Family property.
DAQmxResetDILogicFamily resets the Digital Input >> Logic Family property.
Get/Set/Reset DI_DataXferMech

int32 __CFUNC DAQmxGetDIDataXferMech(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDIDataXferMech(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDIDataXferMech(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDIDataXferMech gets the Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxSetDIDataXferMech sets the Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxResetDIDataXferMech resets the Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.
**Get/Set/Reset DI_DataXferReqCond**

```c
int32 __CFUNC DAQmxGetDIDataXferReqCond(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDIDataXferReqCond(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDIDataXferReqCond(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetDIDataXferReqCond gets the [Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition] property.

DAQmxSetDIDataXferReqCond sets the [Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition] property.

DAQmxResetDIDataXferReqCond resets the [Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition] property.
Get/Set/Reset DI_MemMapEnable

int32 __CFUNC DAQmxGetDIMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDIMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDIMemMapEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDIMemMapEnable gets the **Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable** property.

DAQmxSetDIMemMapEnable sets the **Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable** property.

DAQmxResetDIMemMapEnable resets the **Digital Input >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable** property.
Get/Set/Reset DI_AcquireOn

int32 __CFUNC DAQmxGetDIAcquireOn(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDIAcquireOn(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDIAcquireOn(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetDI AcquireOn gets the [Digital Input >> General Properties >> Advanced >> Acquire On](#) property.

DAQmxSetDI AcquireOn sets the [Digital Input >> General Properties >> Advanced >> Acquire On](#) property.

DAQmxResetDI AcquireOn resets the [Digital Input >> General Properties >> Advanced >> Acquire On](#) property.
Get/Set/Reset DO_OutputDriveType

int32 __CFUNC DAQmxGetDOOutputDriveType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDOOutputDriveType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDOOutputDriveType(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetDOOutputDriveType gets the [Digital Output >> Output Drive Type](#) property.

DAQmxSetDOOutputDriveType sets the [Digital Output >> Output Drive Type](#) property.

DAQmxResetDOOutputDriveType resets the [Digital Output >> Output Drive Type](#) property.
Get/Set/Reset DO_InvertLines

int32 __CFUNC DAQmxGetDOInvertLines(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDOInvertLines(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDOInvertLines(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOInvertLines gets the Digital Output >> Invert Lines property.
DAQmxSetDOInvertLines sets the Digital Output >> Invert Lines property.
DAQmxResetDOInvertLines resets the Digital Output >> Invert Lines property.
Get/Set/Reset DO_NumLines

int32 __CFUNC DAQmxGetDONumLines(TaskHandle taskHandle, const char channel[], uInt32 *data);}
Purpose

DAQmxGetDONumLines gets the Digital Output >> Number of Lines property.
Get/Set/Reset DO_Tristate

int32 __CFUNC DAQmxGetDOTristate(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDOTristate(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDOTristate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOTristate gets the Digital Output >> Tristate property.
DAQmxSetDOTristate sets the Digital Output >> Tristate property.
DAQmxResetDOTristate resets the Digital Output >> Tristate property.
Get/Set/Reset DO_LineStates_StartState

int32 __CFUNC DAQmxGetDOLineStatesStartState(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDOLineStatesStartState(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDOLineStatesStartState(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOLineStatesStartState gets the Digital Output >> Line States >> Start State property.
DAQmxSetDOLineStatesStartState sets the Digital Output >> Line States >> Start State property.
DAQmxResetDOLineStatesStartState resets the Digital Output >> Line States >> Start State property.
Get/Set/Reset DO_LineStates_PausedState

int32 __CFUNC DAQmxGetDOLineStatesPausedState(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDOLineStatesPausedState(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDOLineStatesPausedState(TaskHandle taskHandle, const char channel[]);
Purpose


DAQmxSetDOLineStatesPausedState sets the Digital Output >> Line States >> Paused State property.

Get/Set/Reset DO_LineStates_DoneState

int32 __CFUNC DAQmxGetDOLineStatesDoneState(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDOLineStatesDoneState(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDOLineStatesDoneState(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOLineStatesDoneState gets the Digital Output >> Line States >> Done State property.
DAQmxSetDOLineStatesDoneState sets the Digital Output >> Line States >> Done State property.
DAQmxResetDOLineStatesDoneState resets the Digital Output >> Line States >> Done State property.
Get/Set/Reset DO_LogicFamily

int32 __CFUNC DAQmxGetDOLogicFamily(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDOLogicFamily(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDOLogicFamily(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOLogicFamily gets the Digital Output Logic Family property.
DAQmxSetDOLogicFamily sets the Digital Output Logic Family property.
DAQmxResetDOLogicFamily resets the Digital Output Logic Family property.
Get/Set/Reset DO_UseOnlyOnBrdMem

int32 __CFUNC DAQmxGetDOUseOnlyOnBrdMem(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDOUseOnlyOnBrdMem(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDOUseOnlyOnBrdMem(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOUseOnlyOnBrdMem gets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory property.

DAQmxSetDOUseOnlyOnBrdMem sets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory property.

DAQmxResetDOUseOnlyOnBrdMem resets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Use Only Onboard Memory property.
Get/Set/Reset DO_DataXferMech

int32 __CFUNC DAQmxGetDODataXferMech(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDODataXferMech(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDODataXferMech(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDODataXferMech gets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxSetDODataXferMech sets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxResetDODataXferMech resets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.
Get/Set/Reset DO_DataXferReqCond

int32 __CFUNC DAQmxGetDODataXferReqCond(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDODataXferReqCond(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDODataXferReqCond(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetDODataXferReqCond gets the [Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition](#) property.

DAQmxSetDODataXferReqCond sets the [Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition](#) property.

DAQmxResetDODataXferReqCond resets the [Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Data Transfer Request Condition](#) property.
Get/Set/Reset DO_MemMapEnable

int32 __CFUNC DAQmxGetDOMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetDOMemMapEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetDOMemMapEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetDOMemMapEnable gets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.

DAQmxSetDOMemMapEnable sets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.

DAQmxResetDOMemMapEnable resets the Digital Output >> General Properties >> Advanced >> Data Transfer and Memory >> Memory Mapping for Programmed IO Enable property.
**Get/Set/Reset DO_GenerateOn**

```c
int32 __CFUNC DAQmxGetDOGenerateOn(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetDOGenerateOn(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetDOGenerateOn(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetDOGenerateOn gets the Digital Output >> General Properties >> Advanced >> Generate On property.

DAQmxSetDOGenerateOn sets the Digital Output >> General Properties >> Advanced >> Generate On property.

DAQmxResetDOGenerateOn resets the Digital Output >> General Properties >> Advanced >> Generate On property.
**Get/Set/Reset CI_Max**

```c
int32 __CFUNC DAQmxGetCIMax(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIMax(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIMax(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetCIMax gets the [Counter Input >> Maximum Value](#) property.

DAQmxSetCIMax sets the [Counter Input >> Maximum Value](#) property.

DAQmxResetCIMax resets the [Counter Input >> Maximum Value](#) property.
Get/Set/Reset CI_Min

int32 __CFUNC DAQmxGetCIMin(TaskHandle taskHandle, const char *channel[], float64 *data);

int32 __CFUNC DAQmxSetCIMin(TaskHandle taskHandle, const char *channel[], float64 data);

int32 __CFUNC DAQmxResetCIMin(TaskHandle taskHandle, const char *channel[]);
Purpose

DAQmxGetCIMin gets the Counter Input >> Minimum Value property.
DAQmxSetCIMin sets the Counter Input >> Minimum Value property.
DAQmxResetCIMin resets the Counter Input >> Minimum Value property.
Get/Set/Reset CI_CustomScaleName

int32 __CFUNC DAQmxGetCIConfigure(TaskHandle taskHandle,
        const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIConfigure(TaskHandle taskHandle,
        const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIConfigure(TaskHandle taskHandle,
        const char channel[]);
Purpose

DAQmxGetCICustomScaleName gets the **Counter Input >> Custom Scale Name** property.
DAQmxSetCICustomScaleName sets the **Counter Input >> Custom Scale Name** property.
DAQmxResetCICustomScaleName resets the **Counter Input >> Custom Scale Name** property.
Get/Set/Reset CI_MeasType

int32 __CFUNC DAQmxGetCIMeasType(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetCIMeasType gets the Counter Input >> Measurement Type property.
Get/Set/Reset CI_Freq_Units

int32 __CFUNC DAQmxGetCIFreqUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIFreqUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIFreqUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIFreqUnits gets the Counter Input >> Frequency >> Units property.
DAQmxSetCIFreqUnits sets the Counter Input >> Frequency >> Units property.
DAQmxResetCIFreqUnits resets the Counter Input >> Frequency >> Units property.
Get/Set/Reset CI_Freq_Term

int32 __CFUNC DAQmxGetCIFreqTerm(TaskHandle taskHandle, const char channel[], char *data, Ulnt32 bufferSize);

int32 __CFUNC DAQmxSetCIFreqTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIFreqTerm(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIFreqTerm gets the Counter Input >> Frequency >> Input Terminal property.

DAQmxSetCIFreqTerm sets the Counter Input >> Frequency >> Input Terminal property.

DAQmxResetCIFreqTerm resets the Counter Input >> Frequency >> Input Terminal property.
Get/Set/Reset CI_Freq_SoringEdge

int32 __CFUNC DAQmxGetCIFreqStartingEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIFreqStartingEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIFreqStartingEdge(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIFreqStartingEdge gets the Counter Input >> Frequency >> Starting Edge property.
DAQmxSetCIFreqStartingEdge sets the Counter Input >> Frequency >> Starting Edge property.
DAQmxResetCIFreqStartingEdge resets the Counter Input >> Frequency >> Starting Edge property.
Get/Set/Reset CI_Freq_MeasMeth

int32 __CFUNC DAQmxGetCIFreqMeasMeth(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIFreqMeasMeth(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIFreqMeasMeth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIFreqMeasMeth gets the Counter Input >> Frequency >> Measurement Specifications >> Method property.

DAQmxSetCIFreqMeasMeth sets the Counter Input >> Frequency >> Measurement Specifications >> Method property.

DAQmxResetCIFreqMeasMeth resets the Counter Input >> Frequency >> Measurement Specifications >> Method property.
Get/Set/Reset CI_Freq_MeasTime

int32 __CFUNC DAQmxGetCIFreqMeasTime(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIFreqMeasTime(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIFreqMeasTime(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIFreqMeasTime gets the Counter Input >> Frequency >> Measurement Specifications >> High Frequency >> Measurement Time property.

DAQmxSetCIFreqMeasTime sets the Counter Input >> Frequency >> Measurement Specifications >> High Frequency >> Measurement Time property.

DAQmxResetCIFreqMeasTime resets the Counter Input >> Frequency >> Measurement Specifications >> High Frequency >> Measurement Time property.
Get/Set/Reset CI_Freq_Div

int32 __CFUNC DAQmxGetCIFreqDiv(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCIFreqDiv(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCIFreqDiv(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIFreqDiv gets the Counter Input >> Frequency >> Measurement Specifications >> Large Range >> Divisor property.

DAQmxSetCIFreqDiv sets the Counter Input >> Frequency >> Measurement Specifications >> Large Range >> Divisor property.

DAQmxResetCIFreqDiv resets the Counter Input >> Frequency >> Measurement Specifications >> Large Range >> Divisor property.
Get/Set/Reset CI_Freq_DigFltr_Enable

int32 __CFUNC DAQmxGetCIFreqDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIFreqDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIFreqDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIFreqDigFltrEnable gets the Counter Input >> Frequency >> Digital Filter >> Enable property.

DAQmxSetCIFreqDigFltrEnable sets the Counter Input >> Frequency >> Digital Filter >> Enable property.

DAQmxResetCIFreqDigFltrEnable resets the Counter Input >> Frequency >> Digital Filter >> Enable property.
Get/Set/Reset CI_Freq_DigFltr_MinPulseWidth

int32 __CFUNC DAQmxGetCIFreqDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIFreqDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIFreqDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIFreqDigFltrMinPulseWidth gets the **Minimum Pulse Width** property.

DAQmxSetCIFreqDigFltrMinPulseWidth sets the **Minimum Pulse Width** property.

DAQmxResetCIFreqDigFltrMinPulseWidth resets the **Minimum Pulse Width** property.
Get/Set/Reset CI_Freq_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCIFreqDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIFreqDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIFreqDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIFreqDigFltrTimebaseSrc gets the Counter Input >> Frequency >> Digital Filter >> Timebase >> Source property.

DAQmxSetCIFreqDigFltrTimebaseSrc sets the Counter Input >> Frequency >> Digital Filter >> Timebase >> Source property.

DAQmxResetCIFreqDigFltrTimebaseSrc resets the Counter Input >> Frequency >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset CI_Freq_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetCIFreqDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIFreqDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIFreqDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIFreqDigFltrTimebaseRate gets the \textit{Counter Input \& Frequency \& Digital Filter} \texttt{Timebase >> Rate} property.

DAQmxSetCIFreqDigFltrTimebaseRate sets the \textit{Counter Input \& Frequency \& Digital Filter} \texttt{Timebase >> Rate} property.

DAQmxResetCIFreqDigFltrTimebaseRate resets the \textit{Counter Input \& Frequency \& Digital Filter} \texttt{Timebase >> Rate} property.
Get/Set/Reset CI_Freq_DigSync_Enable

int32 __CFUNC DAQmxGetCIFreqDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIFreqDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIFreqDigSyncEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIFreqDigSyncEnable gets the property.

DAQmxSetCIFreqDigSyncEnable sets the property.

DAQmxResetCIFreqDigSyncEnable resets the property.
Get/Set/Reset CI_Period_Units

int32 __CFUNC DAQmxGetCIPeriodUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIPeriodUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIPeriodUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodUnits gets the Counter Input >> Period >> Units property.
DAQmxSetCIPeriodUnits sets the Counter Input >> Period >> Units property.
DAQmxResetCIPeriodUnits resets the Counter Input >> Period >> Units property.
Get/Set/Reset CI_Period_Term

int32 __CFUNC DAQmxGetCIPeriodTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIPeriodTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIPeriodTerm(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodTerm gets the Counter Input >> Period >> Input Terminal property.

DAQmxSetCIPeriodTerm sets the Counter Input >> Period >> Input Terminal property.

DAQmxResetCIPeriodTerm resets the Counter Input >> Period >> Input Terminal property.
Get/Set/Reset CI_Period StartingEdge

int32 __CFUNC DAQmxGetCIPeriodStartingEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIPeriodStartingEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIPeriodStartingEdge(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodStartingEdge gets the Counter Input >> Period >> Starting Edge property.
DAQmxSetCIPeriodStartingEdge sets the Counter Input >> Period >> Starting Edge property.
DAQmxResetCIPeriodStartingEdge resets the Counter Input >> Period >> Starting Edge property.
Get/Set/Reset CI_Period_MeasMeth

int32 __CFUNC DAQmxGetCIPeriodMeasMeth(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIPeriodMeasMeth(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIPeriodMeasMeth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodMeasMeth gets the Counter Input >> Period >> Measurement Specifications >> Method property.

DAQmxSetCIPeriodMeasMeth sets the Counter Input >> Period >> Measurement Specifications >> Method property.

DAQmxResetCIPeriodMeasMeth resets the Counter Input >> Period >> Measurement Specifications >> Method property.
Get/Set/Reset CI_Period_MeasTime

int32 __CFUNC DAQmxGetCIPeriodMeasTime(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIPeriodMeasTime(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIPeriodMeasTime(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodMeasTime gets the Counter Input >> Period >> Measurement Specifications >> High Frequency >> Measurement Time property.

DAQmxSetCIPeriodMeasTime sets the Counter Input >> Period >> Measurement Specifications >> High Frequency >> Measurement Time property.

DAQmxResetCIPeriodMeasTime resets the Counter Input >> Period >> Measurement Specifications >> High Frequency >> Measurement Time property.
Get/Set/Reset CI_Period_Div

int32 __CFUNC DAQmxGetCIPeriodDiv(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCIPeriodDiv(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCIPeriodDiv(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodDiv gets the Counter Input >> Period >> Measurement Specifications >> Large Range >> Divisor property.

DAQmxSetCIPeriodDiv sets the Counter Input >> Period >> Measurement Specifications >> Large Range >> Divisor property.

DAQmxResetCIPeriodDiv resets the Counter Input >> Period >> Measurement Specifications >> Large Range >> Divisor property.
Get/Set/Reset CI_Period_DigFltr_Enable

```c
int32 __CFUNC DAQmxGetCIPeriodDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIPeriodDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIPeriodDigFltrEnable(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCIPeriodDigFltrEnable gets the **Counter Input >> Period >> Digital Filter >> Enable** property.

DAQmxSetCIPeriodDigFltrEnable sets the **Counter Input >> Period >> Digital Filter >> Enable** property.

DAQmxResetCIPeriodDigFltrEnable resets the **Counter Input >> Period >> Digital Filter >> Enable** property.
Get/Set/Reset CI_Period_DigFltr_MinPulseWidth

int32 __CFUNC DAQmxGetCIPeriodDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIPeriodDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIPeriodDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodDigFltrMinPulseWidth gets the Counter Input >> Period >> Digital Filter >> Minimum Pulse Width property.

DAQmxSetCIPeriodDigFltrMinPulseWidth sets the Counter Input >> Period >> Digital Filter >> Minimum Pulse Width property.

DAQmxResetCIPeriodDigFltrMinPulseWidth resets the Counter Input >> Period >> Digital Filter >> Minimum Pulse Width property.
Get/Set/Reset CI_Period_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCIPeriodDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIPeriodDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIPeriodDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPeriodDigFltrTimebaseSrc gets the Counter Input >> Period >> Digital Filter >> Timebase >> Source property.

DAQmxSetCIPeriodDigFltrTimebaseSrc sets the Counter Input >> Period >> Digital Filter >> Timebase >> Source property.

DAQmxResetCIPeriodDigFltrTimebaseSrc resets the Counter Input >> Period >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset CI_Period_DigFltr_TimebaseRate

```c
int32 __CFUNC DAQmxGetCIPeriodDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIPeriodDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIPeriodDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCIPeriodDigFltrTimebaseRate gets the property.

DAQmxSetCIPeriodDigFltrTimebaseRate sets the property.

DAQmxResetCIPeriodDigFltrTimebaseRate resets the property.
**Get/Set/Reset CI_Period_DigSync_Enable**

```c
int32 __CFUNC DAQmxGetCIPeriodDigSyncEnable(TaskHandle taskHandle,
                                           const char channel[],
                                           bool32 *data);

int32 __CFUNC DAQmxSetCIPeriodDigSyncEnable(TaskHandle taskHandle,
                                           const char channel[],
                                           bool32 data);

int32 __CFUNC DAQmxResetCIPeriodDigSyncEnable(TaskHandle taskHandle,
                                           const char channel[]);
```
Purpose

DAQmxGetCIPeriodDigSyncEnable gets the Counter Input >> Period >> Digital Synchronization >> Enable property.

DAQmxSetCIPeriodDigSyncEnable sets the Counter Input >> Period >> Digital Synchronization >> Enable property.

DAQmxResetCIPeriodDigSyncEnable resets the Counter Input >> Period >> Digital Synchronization >> Enable property.
Get/Set/Reset CI_CountEdges_Term

int32 __CFUNC DAQmxGetCICountEdgesTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCICountEdgesTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCICountEdgesTerm(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesTerm gets the Counter Input >> Count Edges >> Input Terminal property.
DAQmxSetCICountEdgesTerm sets the Counter Input >> Count Edges >> Input Terminal property.
DAQmxResetCICountEdgesTerm resets the Counter Input >> Count Edges >> Input Terminal property.
Get/Set/Reset CI_CountEdges_Dir

int32 __CFUNC DAQmxGetCICountEdgesDir(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCICountEdgesDir(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCICountEdgesDir(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesDir gets the property.

DAQmxSetCICountEdgesDir sets the property.

DAQmxResetCICountEdgesDir resets the property.
Get/Set/Reset CI_CountEdges_DirTerm

int32 __CFUNC DAQmxGetCICountEdgesDirTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCICountEdgesDirTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCICountEdgesDirTerm(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesDirTerm gets the Counter Input >> Count Edges >> Count Direction >> Terminal property.

DAQmxSetCICountEdgesDirTerm sets the Counter Input >> Count Edges >> Count Direction >> Terminal property.

DAQmxResetCICountEdgesDirTerm resets the Counter Input >> Count Edges >> Count Direction >> Terminal property.
**Get/Set/Reset**

**CI_CountEdges_CountDir_DigFltr_Enable**

```c
int32 __CFUNC DAQmxGetCICountEdgesCountDirDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCICountEdgesCountDirDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCICountEdgesCountDirDigFltrEnable(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCICountEdgesCountDirDigFltrEnable gets the `Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Enable` property.

DAQmxSetCICountEdgesCountDirDigFltrEnable sets the `Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Enable` property.

DAQmxResetCICountEdgesCountDirDigFltrEnable resets the `Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Enable` property.
Get/Set/Reset

CI_CountEdges_CountDir_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetCICountEdgesCountDirDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
    DAQmxSetCICountEdgesCountDirDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCICountEdgesCountDirDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesCountDirDigFltrMinPulseWidth gets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Minimum Pulse Width property.

DAQmxSetCICountEdgesCountDirDigFltrMinPulseWidth sets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Minimum Pulse Width property.

DAQmxResetCICountEdgesCountDirDigFltrMinPulseWidth resets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Minimum Pulse Width property.
Get/Set/Reset
CI_CountEdges_CountDir_DigFltr_TimebaseSrc

int32 __CFUNC
    DAQmxGetCICountEdgesCountDirDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetCICountEdgesCountDirDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC
    DAQmxResetCICountEdgesCountDirDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesCountDirDigFltrTimebaseSrc gets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Source property.

DAQmxSetCICountEdgesCountDirDigFltrTimebaseSrc sets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Source property.

DAQmxResetCICountEdgesCountDirDigFltrTimebaseSrc resets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset

CI_CountEdges_CountDir_DigFltr_TimebaseRate

int32 __CFUNC
    DAQmxGetCICountEdgesCountDirDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
    DAQmxSetCICountEdgesCountDirDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCICountEdgesCountDirDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesCountDirDigFltrTimebaseRate gets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Rate property.

DAQmxSetCICountEdgesCountDirDigFltrTimebaseRate sets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Rate property.

DAQmxResetCICountEdgesCountDirDigFltrTimebaseRate resets the Counter Input >> Count Edges >> Count Direction >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset

CI_CountEdges_CountDir_DigSync_Enable

```c
int32 __CFUNC
    DAQmxGetCICountEdgesCountDirDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC
    DAQmxSetCICountEdgesCountDirDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC
    DAQmxResetCICountEdgesCountDirDigSyncEnable(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCICountEdgesCountDirDigSyncEnable gets the Counter Input >> Count Edges >> Count Direction >> Digital Synchronization >> Enable property.

DAQmxSetCICountEdgesCountDirDigSyncEnable sets the Counter Input >> Count Edges >> Count Direction >> Digital Synchronization >> Enable property.

DAQmxResetCICountEdgesCountDirDigSyncEnable resets the Counter Input >> Count Edges >> Count Direction >> Digital Synchronization >> Enable property.
Get/Set/Reset CI_CountEdges_InitialCnt

int32 __CFUNC DAQmxGetCICountEdgesInitialCnt(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCICountEdgesInitialCnt(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCICountEdgesInitialCnt(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesInitialCnt gets the Counter Input >> Count Edges >> Initial Count property.
DAQmxSetCICountEdgesInitialCnt sets the Counter Input >> Count Edges >> Initial Count property.
DAQmxResetCICountEdgesInitialCnt resets the Counter Input >> Count Edges >> Initial Count property.
Get/Set/Reset CI_CountEdges_ActiveEdge

```c
int32 __CFUNC DAQmxGetCICountEdgesActiveEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCICountEdgesActiveEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCICountEdgesActiveEdge(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCICountEdgesActiveEdge gets the property.

DAQmxSetCICountEdgesActiveEdge sets the property.

DAQmxResetCICountEdgesActiveEdge resets the property.
Get/Set/Reset CI_CountEdges_DigFltr_Enable

int32 __CFUNC DAQmxGetCICountEdgesDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCICountEdgesDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCICountEdgesDigFltrEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCICountEdgesDigFltrEnable gets the **Enable** property.

DAQmxSetCICountEdgesDigFltrEnable sets the **Enable** property.

DAQmxResetCICountEdgesDigFltrEnable resets the **Enable** property.
Get/Set/Reset

**CI_CountEdges_DigFltr_MinPulseWidth**

```c
int32 __CFUNC DAQmxGetCICountEdgesDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCICountEdgesDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCICountEdgesDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCICountEdgesDigFltrMinPulseWidth gets the **Counter Input >> Count Edges >> Digital Filter >> Minimum Pulse Width** property.

DAQmxSetCICountEdgesDigFltrMinPulseWidth sets the **Counter Input >> Count Edges >> Digital Filter >> Minimum Pulse Width** property.

DAQmxResetCICountEdgesDigFltrMinPulseWidth resets the **Counter Input >> Count Edges >> Digital Filter >> Minimum Pulse Width** property.
Get/Set/Reset CI_CountEdges_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCICountEdgesDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCICountEdgesDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCICountEdgesDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesDigFltrTimebaseSrc gets the property.

DAQmxSetCICountEdgesDigFltrTimebaseSrc sets the property.

DAQmxResetCICountEdgesDigFltrTimebaseSrc resets the property.
Get/Set/Reset CI_CountEdges_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetCICountEdgesDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCICountEdgesDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCICountEdgesDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesDigFltrTimebaseRate gets the **Counter Input >> Count Edges >> Digital Filter >> Timebase >> Rate** property.

DAQmxSetCICountEdgesDigFltrTimebaseRate sets the **Counter Input >> Count Edges >> Digital Filter >> Timebase >> Rate** property.

DAQmxResetCICountEdgesDigFltrTimebaseRate resets the **Counter Input >> Count Edges >> Digital Filter >> Timebase >> Rate** property.
Get/Set/Reset CI_CountEdges_DigSync_Enable

int32 __CFUNC DAQmxGetCICountEdgesDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCICountEdgesDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCICountEdgesDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICountEdgesDigSyncEnable gets the property.

DAQmxSetCICountEdgesDigSyncEnable sets the property.

DAQmxResetCICountEdgesDigSyncEnable resets the property.
Get/Set/Reset CI_AngEncoder_Units

```c
int32 __CFUNC DAQmxGetCIAngEncoderUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIAngEncoderUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIAngEncoderUnits(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetCIAngEncoderUnits gets the Counter Input >> Position >> Angular Encoder >> Units property.

DAQmxSetCIAngEncoderUnits sets the Counter Input >> Position >> Angular Encoder >> Units property.

DAQmxResetCIAngEncoderUnits resets the Counter Input >> Position >> Angular Encoder >> Units property.
Get/Set/Reset CI_AngEncoder_PulsesPerRev

int32 __CFUNC DAQmxGetCIAngEncoderPulsesPerRev(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCIAngEncoderPulsesPerRev(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCIAngEncoderPulsesPerRev(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIAngEncoderPulsesPerRev gets the Counter Input >> Position >> Angular Encoder >> Pulses Per Revolution property.

DAQmxSetCIAngEncoderPulsesPerRev sets the Counter Input >> Position >> Angular Encoder >> Pulses Per Revolution property.

DAQmxResetCIAngEncoderPulsesPerRev resets the Counter Input >> Position >> Angular Encoder >> Pulses Per Revolution property.
Get/Set/Reset CI_AngEncoder_InititalAngle

int32 __CFUNC DAQmxGetCIAngEncoderInitialAngle(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIAngEncoderInitialAngle(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIAngEncoderInitialAngle(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIAngEncoderInitialAngle gets the Counter Input >> Position >> Angular Encoder >> Initial Angle property.

DAQmxSetCIAngEncoderInitialAngle sets the Counter Input >> Position >> Angular Encoder >> Initial Angle property.

DAQmxResetCIAngEncoderInitialAngle resets the Counter Input >> Position >> Angular Encoder >> Initial Angle property.
Get/Set/Reset **CI_LinEncoder_Units**

```c
int32 __CFUNC DAQmxGetCILinEncoderUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCILinEncoderUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCILinEncoderUnits(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetCILinEncoderUnits gets the Counter >> Position >> Linear Encoder >> Units property.

DAQmxSetCILinEncoderUnits sets the Counter >> Position >> Linear Encoder >> Units property.

DAQmxResetCILinEncoderUnits resets the Counter >> Position >> Linear Encoder >> Units property.
Get/Set/Reset CI_LinEncoder_DistPerPulse

int32 __CFUNC DAQmxGetCILinEncoderDistPerPulse(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCILinEncoderDistPerPulse(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCILinEncoderDistPerPulse(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCILinEncoderDistPerPulse gets the **Counter Input >> Position >> Linear Encoder >> Distance Per Pulse** property.

DAQmxSetCILinEncoderDistPerPulse sets the **Counter Input >> Position >> Linear Encoder >> Distance Per Pulse** property.

DAQmxResetCILinEncoderDistPerPulse resets the **Counter Input >> Position >> Linear Encoder >> Distance Per Pulse** property.
Get/Set/Reset CI_LinEncoder_InitialPos

int32 __CFUNC DAQmxGetCILinEncoderInitialPos(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCILinEncoderInitialPos(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCILinEncoderInitialPos(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCILinEncoderInitialPos gets the Counter Input >> Position >> Linear Encoder >> Initial Position property.

DAQmxSetCILinEncoderInitialPos sets the Counter Input >> Position >> Linear Encoder >> Initial Position property.

DAQmxResetCILinEncoderInitialPos resets the Counter Input >> Position >> Linear Encoder >> Initial Position property.
Get/Set/Reset CI_Encoder_DecodingType

int32 __CFUNC DAQmxGetCIEncoderDecodingType(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIEncoderDecodingType(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIEncoderDecodingType(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderDecodingType gets the Counter Input >> Position >> Decoding Type property.

DAQmxSetCIEncoderDecodingType sets the Counter Input >> Position >> Decoding Type property.

DAQmxResetCIEncoderDecodingType resets the Counter Input >> Position >> Decoding Type property.
Get/Set/Reset CI_Encoder_AInputTerm

int32 __CFUNC DAQmxGetCIEncoderAInputTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIEncoderAInputTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIEncoderAInputTerm(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIEncoderAInputTerm gets the Counter Input >> Position >> A Input >> Terminal property.

DAQmxSetCIEncoderAInputTerm sets the Counter Input >> Position >> A Input >> Terminal property.

DAQmxResetCIEncoderAInputTerm resets the Counter Input >> Position >> A Input >> Terminal property.
Get/Set/Reset CI_Encoder_AInput_DigFltr_Enable

int32 __CFUNC DAQmxGetCIEncoderAInputDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderAInputDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderAInputDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCEncoderAInputDigFltrEnable gets the **Counter Input >> Position >> A Input >> Digital Filter >> Enable** property.

DAQmxSetCEncoderAInputDigFltrEnable sets the **Counter Input >> Position >> A Input >> Digital Filter >> Enable** property.

DAQmxResetCEncoderAInputDigFltrEnable resets the **Counter Input >> Position >> A Input >> Digital Filter >> Enable** property.
Get/Set/Reset

CI_Encoder_AInput_DigFltr_MinPulseWidth

```c
int32 __CFUNC
   DAQmxGetCIEncoderAInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
   DAQmxSetCIEncoderAInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
   DAQmxResetCIEncoderAInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCIEncoderAInputDigFltrMinPulseWidth gets the \texttt{Counter Input >> Position >> A Input >> Digital Filter >> Minimum Pulse Width} property.

DAQmxSetCIEncoderAInputDigFltrMinPulseWidth sets the \texttt{Counter Input >> Position >> A Input >> Digital Filter >> Minimum Pulse Width} property.

DAQmxResetCIEncoderAInputDigFltrMinPulseWidth resets the \texttt{Counter Input >> Position >> A Input >> Digital Filter >> Minimum Pulse Width} property.
Get/Set/Reset
CI_Encoder_AInput_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCIEncoderAInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIEncoderAInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC
    DAQmxResetCIEncoderAInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderAInputDigFltrTimebaseSrc gets the property.

DAQmxSetCIEncoderAInputDigFltrTimebaseSrc sets the property.

DAQmxResetCIEncoderAInputDigFltrTimebaseSrc resets the property.
Get/Set/Reset

CI_Encoder_AInput_DigFltr_TimebaseRate

int32 __CFUNC
    DAQmxGetCIEncoderAInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIEncoderAInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCIEncoderAInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderAInputDigFltrTimebaseRate gets the `Counter Input >> Position >> A Input >> Digital Filter >> Timebase >> Rate` property.

DAQmxSetCIEncoderAInputDigFltrTimebaseRate sets the `Counter Input >> Position >> A Input >> Digital Filter >> Timebase >> Rate` property.

DAQmxResetCIEncoderAInputDigFltrTimebaseRate resets the `Counter Input >> Position >> A Input >> Digital Filter >> Timebase >> Rate` property.
Get/Set/Reset CI_Encoder_AInput_DigSync_Enable

int32 __CFUNC DAQmxGetCIEncoderAInputDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderAInputDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderAInputDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderAInputDigSyncEnable gets the Digital Synchronization >> Enable property.

DAQmxSetCIEncoderAInputDigSyncEnable sets the Digital Synchronization >> Enable property.

DAQmxResetCIEncoderAInputDigSyncEnable resets the Digital Synchronization >> Enable property.
Get/Set/Reset CI_Encoder_BInputTerm

int32 __CFUNC DAQmxGetCIEncoderBInputTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIEncoderBInputTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIEncoderBInputTerm(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIEncoderBInputTerm gets the **Counter Input >> Position >> B Input >> Terminal** property.

DAQmxSetCIEncoderBInputTerm sets the **Counter Input >> Position >> B Input >> Terminal** property.

DAQmxResetCIEncoderBInputTerm resets the **Counter Input >> Position >> B Input >> Terminal** property.
Get/Set/Reset CI_Encoder_BInput_DigFltr_Enable

int32 __CFUNC DAQmxGetCIEncoderBInputDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderBInputDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderBInputDigFltrEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIEncoderBInputDigFltrEnable gets the `Counter Input >> Position >> B Input >> Digital Filter >> Enable` property.

DAQmxSetCIEncoderBInputDigFltrEnable sets the `Counter Input >> Position >> B Input >> Digital Filter >> Enable` property.

DAQmxResetCIEncoderBInputDigFltrEnable resets the `Counter Input >> Position >> B Input >> Digital Filter >> Enable` property.
Get/Set/Reset

CI_Encoder_BInput_DigFltr_MinPulseWidth

```c
int32 __CFUNC
    DAQmxGetCIEncoderBInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
    DAQmxSetCIEncoderBInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCIEncoderBInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCIEncoderBInputDigFltrMinPulseWidth gets the **Counter Input >> Position >> B Input >> Digital Filter >> Minimum Pulse Width** property.

DAQmxSetCIEncoderBInputDigFltrMinPulseWidth sets the **Counter Input >> Position >> B Input >> Digital Filter >> Minimum Pulse Width** property.

DAQmxResetCIEncoderBInputDigFltrMinPulseWidth resets the **Counter Input >> Position >> B Input >> Digital Filter >> Minimum Pulse Width** property.
Get/Set/Reset
CI_Encoder_BInput_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCIEncoderBInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIEncoderBInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC
    DAQmxResetCIEncoderBInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderBInputDigFltrTimebaseSrc gets the Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Source property.

DAQmxSetCIEncoderBInputDigFltrTimebaseSrc sets the Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Source property.

DAQmxResetCIEncoderBInputDigFltrTimebaseSrc resets the Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset
CI_Encoder_BInput_DigFltr_TimebaseRate

int32 __CFUNC
    DAQmxGetCIEncoderBInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIEncoderBInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCIEncoderBInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderBInputDigFltrTimebaseRate gets the Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Rate property.

DAQmxSetCIEncoderBInputDigFltrTimebaseRate sets the Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Rate property.

DAQmxResetCIEncoderBInputDigFltrTimebaseRate resets the Counter Input >> Position >> B Input >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset CI_Encoder_BInput_DigSync_Enable

int32 __CFUNC DAQmxGetCIEncoderBInputDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderBInputDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderBInputDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderBInputDigSyncEnable gets the Counter Input >> Position >> B Input >> Digital Synchronization >> Enable property.

DAQmxSetCIEncoderBInputDigSyncEnable sets the Counter Input >> Position >> B Input >> Digital Synchronization >> Enable property.

DAQmxResetCIEncoderBInputDigSyncEnable resets the Counter Input >> Position >> B Input >> Digital Synchronization >> Enable property.
Get/Set/Reset CI_Encoder_ZInputTerm

int32 __CFUNC DAQmxGetCIEncoderZInputTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIEncoderZInputTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIEncoderZInputTerm(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIEncoderZInputTerm gets the Counter Input >> Position >> Z Input >> Terminal property.

DAQmxSetCIEncoderZInputTerm sets the Counter Input >> Position >> Z Input >> Terminal property.

DAQmxResetCIEncoderZInputTerm resets the Counter Input >> Position >> Z Input >> Terminal property.
Get/Set/Reset CI_Encoder_ZInput_DigFltr_Enable

int32 __CFUNC DAQmxGetCIEncoderZInputDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderZInputDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderZInputDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderZInputDigFltrEnable gets the Countter Input >> Position >> Z Input >> Digital Filter >> Enable property.

DAQmxSetCIEncoderZInputDigFltrEnable sets the Countter Input >> Position >> Z Input >> Digital Filter >> Enable property.

DAQmxResetCIEncoderZInputDigFltrEnable resets the Countter Input >> Position >> Z Input >> Digital Filter >> Enable property.
Get/Set/Reset

CI_Encoder_ZInput_DigFltr_MinPulseWidth

```c
int32 __CFUNC
    DAQmxGetCIEncoderZInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);
```

```c
int32 __CFUNC
    DAQmxSetCIEncoderZInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);
```

```c
int32 __CFUNC
    DAQmxResetCIEncoderZInputDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCIEncoderZInputDigFltrMinPulseWidth gets the property.

DAQmxSetCIEncoderZInputDigFltrMinPulseWidth sets the property.

DAQmxResetCIEncoderZInputDigFltrMinPulseWidth resets the property.
Get/Set/Reset

CI_Encoder_ZInput_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCIEncoderZInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIEncoderZInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC
    DAQmxResetCIEncoderZInputDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderZInputDigFltrTimebaseSrc gets the Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Source property.

DAQmxSetCIEncoderZInputDigFltrTimebaseSrc sets the Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Source property.

DAQmxResetCIEncoderZInputDigFltrTimebaseSrc resets the Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset

CI_Encoder_ZInput_DigFltr_TimebaseRate

int32 __CFUNC
    DAQmxGetCIEncoderZInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIEncoderZInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCIEncoderZInputDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIEncoderZInputDigFltrTimebaseRate gets the Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Rate property.

DAQmxSetCIEncoderZInputDigFltrTimebaseRate sets the Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Rate property.

DAQmxResetCIEncoderZInputDigFltrTimebaseRate resets the Counter Input >> Position >> Z Input >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset CI_Encoder_ZInput_DigSync_Enable

int32 __CFUNC DAQmxGetCIEncoderZInputDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderZInputDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderZInputDigSyncEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIEncoderZInputDigSyncEnable gets the [Counter Input >> Position >> Z Input >> Digital Synchronization >> Enable](#) property.

DAQmxSetCIEncoderZInputDigSyncEnable sets the [Counter Input >> Position >> Z Input >> Digital Synchronization >> Enable](#) property.

DAQmxResetCIEncoderZInputDigSyncEnable resets the [Counter Input >> Position >> Z Input >> Digital Synchronization >> Enable](#) property.
Get/Set/Reset CI_Encoder_ZIndexEnable

int32 __CFUNC DAQmxGetCIEncoderZIndexEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIEncoderZIndexEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIEncoderZIndexEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderZIndexEnable gets the Counter Input >> Position >> Z Index Enable property. DAQmxSetCIEncoderZIndexEnable sets the Counter Input >> Position >> Z Index Enable property. DAQmxResetCIEncoderZIndexEnable resets the Counter Input >> Position >> Z Index Enable property.
Get/Set/Reset CI_Encoder_ZIndexVal

int32 __CFUNC DAQmxGetCIEncoderZIndexVal(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIEncoderZIndexVal(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIEncoderZIndexVal(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderZIndexVal gets the Counter Input >> Position >> Z Index Value property.
DAQmxSetCIEncoderZIndexVal sets the Counter Input >> Position >> Z Index Value property.
DAQmxResetCIEncoderZIndexVal resets the Counter Input >> Position >> Z Index Value property.
Get/Set/Reset CI_Encoder_ZIndexPhase

int32 __CFUNC DAQmxGetCIEncoderZIndexPhase(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIEncoderZIndexPhase(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIEncoderZIndexPhase(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIEncoderZIndexPhase gets the Counter >> Position >> Z Index Phase property. DAQmxSetCIEncoderZIndexPhase sets the Counter >> Position >> Z Index Phase property. DAQmxResetCIEncoderZIndexPhase resets the Counter >> Position >> Z Index Phase property.
Get/Set/Reset CI_PulseWidth_Units

int32 __CFUNC DAQmxGetCIPulseWidthUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIPulseWidthUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIPulseWidthUnits(TaskHandle taskHandle, const char channel[]);


Purpose

DAQmxGetCIPulseWidthUnits gets the Counter Input >> Pulse Width >> Units property.
DAQmxSetCIPulseWidthUnits sets the Counter Input >> Pulse Width >> Units property.
DAQmxResetCIPulseWidthUnits resets the Counter Input >> Pulse Width >> Units property.
Get/Set/Reset CI_PulseWidth_Term

int32 __CFUNC DAQmxGetCIPulseWidthTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIPulseWidthTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIPulseWidthTerm(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPulseWidthTerm gets the "Counter Input >> Pulse Width >> Input Terminal" property.
DAQmxSetCIPulseWidthTerm sets the "Counter Input >> Pulse Width >> Input Terminal" property.
DAQmxResetCIPulseWidthTerm resets the "Counter Input >> Pulse Width >> Input Terminal" property.
Get/Set/Reset CI_PulseWidth_StartingEdge

int32 __CFUNC DAQmxGetCIPulseWidthStartingEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIPulseWidthStartingEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIPulseWidthStartingEdge(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPulseWidthStartingEdge gets the `Counter Input >> Pulse Width >> Starting Edge` property.

DAQmxSetCIPulseWidthStartingEdge sets the `Counter Input >> Pulse Width >> Starting Edge` property.

DAQmxResetCIPulseWidthStartingEdge resets the `Counter Input >> Pulse Width >> Starting Edge` property.
Get/Set/Reset CI_PulseWidth_DigFltr_Enable

int32 __CFUNC DAQmxGetCIPulseWidthDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIPulseWidthDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIPulseWidthDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPulseWidthDigFltrEnable gets the Enable property.

DAQmxSetCIPulseWidthDigFltrEnable sets the Enable property.

DAQmxResetCIPulseWidthDigFltrEnable resets the Enable property.
Get/Set/Reset
CI_PulseWidth_DigFltr_MinPulseWidth

int32 __CFUNC DAQmxGetCIPulseWidthDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIPulseWidthDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIPulseWidthDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPulseWidthDigFltrMinPulseWidth gets the Counter Input >> Pulse Width >> Digital Filter >> Minimum Pulse Width property.

DAQmxSetCIPulseWidthDigFltrMinPulseWidth sets the Counter Input >> Pulse Width >> Digital Filter >> Minimum Pulse Width property.

DAQmxResetCIPulseWidthDigFltrMinPulseWidth resets the Digital Filter >> Minimum Pulse Width property.
Get/Set/Reset CI_PulseWidth_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCIPulseWidthDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCIPulseWidthDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIPulseWidthDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPulseWidthDigFltrTimebaseSrc gets the Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Source property.

DAQmxSetCIPulseWidthDigFltrTimebaseSrc sets the Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Source property.

DAQmxResetCIPulseWidthDigFltrTimebaseSrc resets the Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset CI_PulseWidth_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetCIPulseWidthDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCIPulseWidthDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCIPulseWidthDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIPulseWidthDigFltrTimebaseRate gets the Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Rate property.

DAQmxSetCIPulseWidthDigFltrTimebaseRate sets the Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Rate property.

DAQmxResetCIPulseWidthDigFltrTimebaseRate resets the Counter Input >> Pulse Width >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset CI_PulseWidth_DigSync_Enable

int32 __CFUNC DAQmxGetCIPulseWidthDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIPulseWidthDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIPulseWidthDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIPulseWidthDigSyncEnable gets the Counter Input >> Pulse Width >> Digital Synchronization >> Enable property.

DAQmxSetCIPulseWidthDigSyncEnable sets the Counter Input >> Pulse Width >> Digital Synchronization >> Enable property.

DAQmxResetCIPulseWidthDigSyncEnable resets the Counter Input >> Pulse Width >> Digital Synchronization >> Enable property.
Get/Set/Reset CI_TwoEdgeSep_Units

```c
int32 __CFUNC DAQmxGetCITwoEdgeSepUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCITwoEdgeSepUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCITwoEdgeSepUnits(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetCITwoEdgeSepUnits gets the Counter Input >> Two Edge Separation >> Units property.

DAQmxSetCITwoEdgeSepUnits sets the Counter Input >> Two Edge Separation >> Units property.

DAQmxResetCITwoEdgeSepUnits resets the Counter Input >> Two Edge Separation >> Units property.
Get/Set/Reset CI_TwoEdgeSep_FirstTerm

int32 __CFUNC DAQmxGetCITwoEdgeSepFirstTerm(TaskHandle taskHandle,
const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCITwoEdgeSepFirstTerm(TaskHandle taskHandle,
const char channel[], const char *data);

int32 __CFUNC DAQmxResetCITwoEdgeSepFirstTerm(TaskHandle taskHandle,
const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepFirstTerm gets the Counter Input >> Two Edge Separation >> First >> Input Terminal property.

DAQmxSetCITwoEdgeSepFirstTerm sets the Counter Input >> Two Edge Separation >> First >> Input Terminal property.

DAQmxResetCITwoEdgeSepFirstTerm resets the Counter Input >> Two Edge Separation >> First >> Input Terminal property.
Get/Set/Reset CI_TwoEdgeSep_FirstEdge

int32 __CFUNC DAQmxGetCITwoEdgeSepFirstEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCITwoEdgeSepFirstEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCITwoEdgeSepFirstEdge(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCITwoEdgeSepFirstEdge gets the Counter Input >> Two Edge Separation >> First >> Edge property.

DAQmxSetCITwoEdgeSepFirstEdge sets the Counter Input >> Two Edge Separation >> First >> Edge property.

DAQmxResetCITwoEdgeSepFirstEdge resets the Counter Input >> Two Edge Separation >> First >> Edge property.
Get/Set/Reset CI_TwoEdgeSep_First_DigFltr_Enable

int32 __CFUNC DAQmxGetCITwoEdgeSepFirstDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCITwoEdgeSepFirstDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCITwoEdgeSepFirstDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepFirstDigFltrEnable gets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Enable property.

DAQmxSetCITwoEdgeSepFirstDigFltrEnable sets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Enable property.

DAQmxResetCITwoEdgeSepFirstDigFltrEnable resets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Enable property.
Get/Set/Reset
CI_TwoEdgeSep_First_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetCITwoEdgeSepFirstDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
    DAQmxSetCITwoEdgeSepFirstDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCITwoEdgeSepFirstDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepFirstDigFltrMinPulseWidth gets the property.

DAQmxSetCITwoEdgeSepFirstDigFltrMinPulseWidth sets the property.

DAQmxResetCITwoEdgeSepFirstDigFltrMinPulseWidth resets the property.
Get/Set/Reset
CI_TwoEdgeSep_First_DigFltr_TimebaseSrc

int32 __CFUNC
DAQmxGetCITwoEdgeSepFirstDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC
DAQmxSetCITwoEdgeSepFirstDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC
DAQmxResetCITwoEdgeSepFirstDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepFirstDigFltrTimebaseSrc gets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Source property.

DAQmxSetCITwoEdgeSepFirstDigFltrTimebaseSrc sets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Source property.

DAQmxResetCITwoEdgeSepFirstDigFltrTimebaseSrc resets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset
CI_TwoEdgeSep_First_DigFltr_TimebaseRate

int32 __CFUNC
  DAQmxGetCITwoEdgeSepFirstDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
  DAQmxSetCITwoEdgeSepFirstDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
  DAQmxResetCITwoEdgeSepFirstDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepFirstDigFltrTimebaseRate gets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Rate property.

DAQmxSetCITwoEdgeSepFirstDigFltrTimebaseRate sets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Rate property.

DAQmxResetCITwoEdgeSepFirstDigFltrTimebaseRate resets the Counter Input >> Two Edge Separation >> First >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset
CI_TwoEdgeSep_First_DigSync_Enable

int32 __CFUNC DAQmxGetCITwoEdgeSepFirstDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCITwoEdgeSepFirstDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCITwoEdgeSepFirstDigSyncEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCITwoEdgeSepFirstDigSyncEnable gets the property.

DAQmxSetCITwoEdgeSepFirstDigSyncEnable sets the property.

DAQmxResetCITwoEdgeSepFirstDigSyncEnable resets the property.
Get/Set/Reset CI_TwoEdgeSep_SecondTerm

int32 __CFUNC DAQmxGetCITwoEdgeSepSecondTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCITwoEdgeSepSecondTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCITwoEdgeSepSecondTerm(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepSecondTerm gets the Counter Input >> Two Edge Separation >> Second >> Input Terminal property.

DAQmxSetCITwoEdgeSepSecondTerm sets the Counter Input >> Two Edge Separation >> Second >> Input Terminal property.

DAQmxResetCITwoEdgeSepSecondTerm resets the Counter Input >> Two Edge Separation >> Second >> Input Terminal property.
Get/Set/Reset CI_TwoEdgeSep_SecondEdge

int32 __CFUNC DAQmxGetCITwoEdgeSepSecondEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCITwoEdgeSepSecondEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCITwoEdgeSepSecondEdge(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCTwoEdgeSepSecondEdge gets the Counter Input >> Two Edge Separation >> Second >> Edge property.

DAQmxSetCTwoEdgeSepSecondEdge sets the Counter Input >> Two Edge Separation >> Second >> Edge property.

DAQmxResetCTTwoEdgeSepSecondEdge resets the Counter Input >> Two Edge Separation >> Second >> Edge property.
Get/Set/Reset
CI_TwoEdgeSep_Second_DigFltr_Enable

int32 __CFUNC DAQmxGetCITwoEdgeSepSecondDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCITwoEdgeSepSecondDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC
    DAQmxResetCITwoEdgeSepSecondDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepSecondDigFltrEnable gets the Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Enable property.

DAQmxSetCITwoEdgeSepSecondDigFltrEnable sets the Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Enable property.

DAQmxResetCITwoEdgeSepSecondDigFltrEnable resets the Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Enable property.
Get/Set/Reset
CI_TwoEdgeSep_Second_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetCITwoEdgeSepSecondDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
    DAQmxSetCITwoEdgeSepSecondDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCITwoEdgeSepSecondDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepSecondDigFltrMinPulseWidth gets the Counter Input Two Edge Separation Second Digital Filter Minimum Pulse Width property.

DAQmxSetCITwoEdgeSepSecondDigFltrMinPulseWidth sets the Counter Input Two Edge Separation Second Digital Filter Minimum Pulse Width property.

DAQmxResetCITwoEdgeSepSecondDigFltrMinPulseWidth resets the Counter Input Two Edge Separation Second Digital Filter Minimum Pulse Width property.
**Get/Set/Reset**

**CI_TwoEdgeSep_Second_DigFltr_TimebaseSrc**

```c
int32 __CFUNC
    DAQmxGetCITwoEdgeSepSecondDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetCITwoEdgeSepSecondDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC
    DAQmxResetCITwoEdgeSepSecondDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCITwoEdgeSepSecondDigFltrTimebaseSrc gets the **Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Source** property.

DAQmxSetCITwoEdgeSepSecondDigFltrTimebaseSrc sets the **Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Source** property.

DAQmxResetCITwoEdgeSepSecondDigFltrTimebaseSrc resets the **Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Source** property.
Get/Set/Reset
CI_TwoEdgeSep_Second_DigFltr_TimebaseRate

int32 __CFUNC
   DAQmxGetCITwoEdgeSepSecondDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC
   DAQmxSetCITwoEdgeSepSecondDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
   DAQmxResetCITwoEdgeSepSecondDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepSecondDigFltrTimebaseRate gets the **Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Rate** property.

DAQmxSetCITwoEdgeSepSecondDigFltrTimebaseRate sets the **Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Rate** property.

DAQmxResetCITwoEdgeSepSecondDigFltrTimebaseRate resets the **Counter Input >> Two Edge Separation >> Second >> Digital Filter >> Timebase >> Rate** property.
Get/Set/Reset
CI_TwoEdgeSep_Second_DigSync_Enable

int32 __FUNC DAQmxGetCITwoEdgeSepSecondDigSyncEnable(TaskHandle taskHandle, const char channel[], bool *data);

int32 __FUNC DAQmxSetCITwoEdgeSepSecondDigSyncEnable(TaskHandle taskHandle, const char channel[], bool data);

int32 __FUNC
    DAQmxResetCITwoEdgeSepSecondDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCITwoEdgeSepSecondDigSyncEnable gets the Counter Input >> Two Edge Separation >> Second >> Digital Synchronization >> Enable property.

DAQmxSetCITwoEdgeSepSecondDigSyncEnable sets the Counter Input >> Two Edge Separation >> Second >> Digital Synchronization >> Enable property.

DAQmxResetCITwoEdgeSepSecondDigSyncEnable resets the Counter Input >> Two Edge Separation >> Second >> Digital Synchronization >> Enable property.
Get/Set/Reset CI_SemiPeriod_Units

int32 __CFUNC DAQmxGetCISemiPeriodUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCISemiPeriodUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCISemiPeriodUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCISemiPeriodUnits gets the Counter Input >> Semi-Period >> Units property.
DAQmxSetCISemiPeriodUnits sets the Counter Input >> Semi-Period >> Units property.
DAQmxResetCISemiPeriodUnits resets the Counter Input >> Semi-Period >> Units property.
Get/Set/Reset CI_SemiPeriod_Term

```c
int32 __CFUNC DAQmxGetCISemiPeriodTerm(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCISemiPeriodTerm(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCISemiPeriodTerm(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCISemiPeriodTerm gets the Counter Input >> Semi-Period >> Input Terminal property.
DAQmxSetCISemiPeriodTerm sets the Counter Input >> Semi-Period >> Input Terminal property.
DAQmxResetCISemiPeriodTerm resets the Counter Input >> Semi-Period >> Input Terminal property.
Get/Set/Reset CI_SemiPeriod_StartingEdge

```c
int32 __CFUNC DAQmxGetCISemiPeriodStartingEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCISemiPeriodStartingEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCISemiPeriodStartingEdge(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetCISemiPeriodStartingEdge gets the [Counter Input >> Semi-Period >> Starting Edge](#)

DAQmxSetCISemiPeriodStartingEdge sets the [Counter Input >> Semi-Period >> Starting Edge](#)

DAQmxResetCISemiPeriodStartingEdge resets the [Counter Input >> Semi-Period >> Starting Edge](#) property.
Get/Set/Reset CI_SemiPeriod_DigFltr_Enable

int32 __CFUNC DAQmxGetCISemiPeriodDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCISemiPeriodDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCISemiPeriodDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCISemiPeriodDigFltrEnable gets the `Counter Input >> Semi-Period >> Digital Filter >> Enable` property.

DAQmxSetCISemiPeriodDigFltrEnable sets the `Counter Input >> Semi-Period >> Digital Filter >> Enable` property.

DAQmxResetCISemiPeriodDigFltrEnable resets the `Counter Input >> Semi-Period >> Digital Filter >> Enable` property.
Get/Set/Reset

**CI_SemiPeriod_DigFltr_MinPulseWidth**

```c
int32 __CFUNC DAQmxGetCISemiPeriodDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);
```

```c
int32 __CFUNC DAQmxSetCISemiPeriodDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);
```

```c
int32 __CFUNC DAQmxResetCISemiPeriodDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCISemiPeriodDigFltrMinPulseWidth gets the **Counter Input >> Semi-Period >> Digital Filter >> Minimum Pulse Width** property.

DAQmxSetCISemiPeriodDigFltrMinPulseWidth sets the **Counter Input >> Semi-Period >> Digital Filter >> Minimum Pulse Width** property.

DAQmxResetCISemiPeriodDigFltrMinPulseWidth resets the **Counter Input >> Semi-Period >> Digital Filter >> Minimum Pulse Width** property.
Get/Set/Reset CI_SemiPeriod_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCI_SemiPeriod_DigFltr_TimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCI_SemiPeriod_DigFltr_TimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCI_SemiPeriod_DigFltr_TimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCISemiPeriodDigFltrTimebaseSrc gets the Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Source property.

DAQmxSetCISemiPeriodDigFltrTimebaseSrc sets the Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Source property.

DAQmxResetCISemiPeriodDigFltrTimebaseSrc resets the Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset CI_SemiPeriod_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetCISemiPeriodDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCISemiPeriodDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCISemiPeriodDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCISemiPeriodDigFltrTimebaseRate gets the Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Rate property.

DAQmxSetCISemiPeriodDigFltrTimebaseRate sets the Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Rate property.

DAQmxResetCISemiPeriodDigFltrTimebaseRate resets the Counter Input >> Semi-Period >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset CI_SemiPeriod_DigSync_Enable

int32 __CFUNC DAQmxGetCISemiPeriodDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCISemiPeriodDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCISemiPeriodDigSyncEnable(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCISemiPeriodDigSyncEnable gets the Counter Input >> Semi-Period >> Digital Synchronization >> Enable property.

DAQmxSetCISemiPeriodDigSyncEnable sets the Counter Input >> Semi-Period >> Digital Synchronization >> Enable property.

DAQmxResetCISemiPeriodDigSyncEnable resets the Counter Input >> Semi-Period >> Digital Synchronization >> Enable property.
Get/Set/Reset CI_Timestamp_Units

int32 __CFUNC DAQmxGetCITimestampUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCITimestampUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCITimestampUnits(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCITimestampUnits gets the [Counter Input >> Timestamp >> Units](https://example.com) property.

DAQmxSetCITimestampUnits sets the [Counter Input >> Timestamp >> Units](https://example.com) property.

DAQmxResetCITimestampUnits resets the [Counter Input >> Timestamp >> Units](https://example.com) property.
Get/Set/Reset CI_Timestamp_InitialSeconds

int32 __CFUNC DAQmxGetCITimestampInitialSeconds(TaskHandle taskHandle, const char channel[], Uint32 *data);

int32 __CFUNC DAQmxSetCITimestampInitialSeconds(TaskHandle taskHandle, const char channel[], Uint32 data);

int32 __CFUNC DAQmxResetCITimestampInitialSeconds(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCITimestampInitialSeconds gets the `Counter Input >> Timestamp >> Initial Seconds` property.

DAQmxSetCITimestampInitialSeconds sets the `Counter Input >> Timestamp >> Initial Seconds` property.

DAQmxResetCITimestampInitialSeconds resets the `Counter Input >> Timestamp >> Initial Seconds` property.
Get/Set/Reset CI_GPS_SyncMethod

int32 __CFUNC DAQmxGetCIGPSSyncMethod(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIGPSSyncMethod(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIGPSSyncMethod(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIGPSSyncMethod gets the Counter Input >> Timestamp >> GPS >> Synchronization Method property.

DAQmxSetCIGPSSyncMethod sets the Counter Input >> Timestamp >> GPS >> Synchronization Method property.

DAQmxResetCIGPSSyncMethod resets the Counter Input >> Timestamp >> GPS >> Synchronization Method property.
Get/Set/Reset CI_GPS_SyncSrc

int32 __CFUNC DAQmxGetCIGPSSyncSrc(TaskHandle taskHandle, const char channel[], char *data, uint32 bufferSize);

int32 __CFUNC DAQmxSetCIGPSSyncSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCIGPSSyncSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIGPSSyncSrc gets the property.

DAQmxSetCIGPSSyncSrc sets the property.

DAQmxResetCIGPSSyncSrc resets the property.
Get/Set/Reset CI_CtrTimebaseSrc

int32 __CFUNC DAQmxGetCICtrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCICtrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCICtrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCICtrTimebaseSrc gets the property.

DAQmxSetCICtrTimebaseSrc sets the property.

DAQmxResetCICtrTimebaseSrc resets the property.
Get/Set/Reset CI_CtrTimebaseRate

int32 __CFUNC DAQmxGetCICtrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCICtrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCICtrTimebaseRate(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCICtrTimebaseRate gets the Counter Input >> General Properties >> Counter Timebase >> Rate property.

DAQmxSetCICtrTimebaseRate sets the Counter Input >> General Properties >> Counter Timebase >> Rate property.

DAQmxResetCICtrTimebaseRate resets the Counter Input >> General Properties >> Counter Timebase >> Rate property.
Get/Set/Reset CI_CtrTimebaseActiveEdge

```c
int32 __CFUNC DAQmxGetICtrTimebaseActiveEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetICtrTimebaseActiveEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetICtrTimebaseActiveEdge(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCICtrTimebaseActiveEdge gets the Counter Input >> General Properties >> Counter Timebase >> Active Edge property.

DAQmxSetCICtrTimebaseActiveEdge sets the Counter Input >> General Properties >> Counter Timebase >> Active Edge property.

DAQmxResetCICtrTimebaseActiveEdge resets the Counter Input >> General Properties >> Counter Timebase >> Active Edge property.
Get/Set/Reset CI_CtrTimebase_DigFltr_Enable

```c
int32 __CFUNC DAQmxGetCICtrTimebaseDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCICtrTimebaseDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCICtrTimebaseDigFltrEnable(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCICtrTimebaseDigFltrEnable gets the \texttt{Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Enable} property.

DAQmxSetCICtrTimebaseDigFltrEnable sets the \texttt{Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Enable} property.

DAQmxResetCICtrTimebaseDigFltrEnable resets the \texttt{Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Enable} property.
Get/Set/Reset

CI_CtrTimebase_DigFltr_MinPulseWidth

int32 __CFUNC DAQmxGetCICtrTimebaseDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCICtrTimebaseDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCICtrTimebaseDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICtrTimebaseDigFltrMinPulseWidth gets the Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width property.

DAQmxSetCICtrTimebaseDigFltrMinPulseWidth sets the Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width property.

DAQmxResetCICtrTimebaseDigFltrMinPulseWidth resets the Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width property.
Get/Set/Reset CI_CtrTimebase_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCI_CtrTimebaseDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCI_CtrTimebaseDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCI_CtrTimebaseDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICtrTimebaseDigFltrTimebaseSrc gets the property.

DAQmxSetCICtrTimebaseDigFltrTimebaseSrc sets the property.

DAQmxResetCICtrTimebaseDigFltrTimebaseSrc resets the property.
Get/Set/Reset
CI_CtrTimebase_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetCICtrTimebaseDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCICtrTimebaseDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCICtrTimebaseDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCICtrTimebaseDigFltrTimebaseRate gets the [Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Rate](#) property.

DAQmxSetCICtrTimebaseDigFltrTimebaseRate sets the [Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Rate](#) property.

DAQmxResetCICtrTimebaseDigFltrTimebaseRate resets the [Counter Input >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Rate](#) property.
Get/Set/Reset CI_CtrTimebase_DigSync_Enable

int32 __CFUNC DAQmxGetICtrTimebaseDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetICtrTimebaseDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetICtrTimebaseDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICtrTimebaseDigSyncEnable gets the Counter Input >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable property.

DAQmxSetCICtrTimebaseDigSyncEnable sets the Counter Input >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable property.

DAQmxResetCICtrTimebaseDigSyncEnable resets the Counter Input >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable property.
**Get/Set/Reset CI_Count**

```c
int32 __CFUNC DAQmxGetCICount(TaskHandle taskHandle, const char *channel[], uInt32 *data);
```
Purpose

DAQmxGetCICount gets the [Counter Input >> General Properties >> More >> Count](link) property.
Get/Set/Reset CI_OutputState

int32 __CFUNC DAQmxGetCIOutputState(TaskHandle taskHandle, const char *channel[], int32 *data);
Purpose

DAQmxGetCIOoutputState gets the Counter Input >> General Properties >> More >> Output State property.
Get/Set/Reset CI_TCReached

int32 __CFUNC DAQmxGetCITCReached(TaskHandle taskHandle, const char channel[], bool32 *data);
Purpose

DAQmxGetCITCReached gets the Counter Input >> General Properties >> More >> Terminal Count Reached property.
Get/Set/Reset CI_CtrTimebaseMasterTimebaseDiv

int32 __CFUNC DAQmxGetCICtrTimebaseMasterTimebaseDiv(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCICtrTimebaseMasterTimebaseDiv(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCICtrTimebaseMasterTimebaseDiv(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCICtrTimebaseMasterTimebaseDiv gets the Counter Input >> General Properties >> Counter Timebase Master Timebase Divisor property.

DAQmxSetCICtrTimebaseMasterTimebaseDiv sets the Counter Input >> General Properties >> Counter Timebase Master Timebase Divisor property.

DAQmxResetCICtrTimebaseMasterTimebaseDiv resets the Counter Input >> General Properties >> Counter Timebase Master Timebase Divisor property.
Get/Set/Reset CI_DataXferMech

int32 __CFUNC DAQmxGetCIDataXferMech(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCIDataXferMech(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCIDataXferMech(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIDataXferMech gets the Counter Input >> General Properties >> More >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxSetCIDataXferMech sets the Counter Input >> General Properties >> More >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.

DAQmxResetCIDataXferMech resets the Counter Input >> General Properties >> More >> Advanced >> Data Transfer and Memory >> Data Transfer Mechanism property.
Get/Set/Reset CI_NumPossiblyInvalidSamps

int32 __CFUNC DAQmxGetCINumPossiblyInvalidSamps(TaskHandle taskHandle, const char channel[], uInt32 *data);
Purpose

DAQmxGetCINumPossiblyInvalidSamps gets the [Counter Input >> General Properties >> More >> Advanced >> Data Transfer and Memory >> Number Of Possibly Invalid Samples] property.
Get/Set/Reset CI_DupCountPrevent

int32 __CFUNC DAQmxGetCIDupCountPrevent(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCIDupCountPrevent(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCIDupCountPrevent(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCIDupCountPrevent gets the property.

DAQmxSetCIDupCountPrevent sets the property.

DAQmxResetCIDupCountPrevent resets the property.
Get/Set/Reset CI_Prescaler

int32 __CFUNC DAQmxGetCI Prescaler(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCI Prescaler(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCI Prescaler(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCIPrescaler gets the property.

DAQmxSetCIPrescaler sets the property.

DAQmxResetCIPrescaler resets the property.
Get/Set/Reset CO_OutputType

int32 __CFUNC DAQmxGetCOOutputType(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetCOOutputType gets the Counter Output >> Output Type property.
Get/Set/Reset CO_Pulse_IdleState

```c
int32 __CFUNC DAQmxGetCOPulseIdleState(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCOPulseIdleState(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCOPulseIdleState(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCOPulseIdleState gets the Counter Output >> Pulse >> Idle State property.

DAQmxSetCOPulseIdleState sets the Counter Output >> Pulse >> Idle State property.

DAQmxResetCOPulseIdleState resets the Counter Output >> Pulse >> Idle State property.
**Get/Set/Reset CO_Pulse_Term**

```c
int32 __CFUNC DAQmxGetCOPulseTerm(TaskHandle taskHandle, const char *channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCOPulseTerm(TaskHandle taskHandle, const char *channel[], const char *data);

int32 __CFUNC DAQmxResetCOPulseTerm(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCOPulseTerm gets the Counter Output >> Pulse >> Output Terminal property.
DAQmxSetCOPulseTerm sets the Counter Output >> Pulse >> Output Terminal property.
DAQmxResetCOPulseTerm resets the Counter Output >> Pulse >> Output Terminal property.
Get/Set/Reset CO_Pulse_Time_Units

int32 __CFUNC DAQmxGetCOPulseTimeUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCOPulseTimeUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCOPulseTimeUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOPulseTimeUnits gets the Counter Output >> Pulse >> Time >> Units property.

DAQmxSetCOPulseTimeUnits sets the Counter Output >> Pulse >> Time >> Units property.

DAQmxResetCOPulseTimeUnits resets the Counter Output >> Pulse >> Time >> Units property.
Get/Set/Reset CO_Pulse_HighTime

int32 __CFUNC DAQmxGetCOPulseHighTime(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOPulseHighTime(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCOPulseHighTime(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCOPulseHighTime gets the Counter Output >> Pulse >> Time >> High Time property.

DAQmxSetCOPulseHighTime sets the Counter Output >> Pulse >> Time >> High Time property.

DAQmxResetCOPulseHighTime resets the Counter Output >> Pulse >> Time >> High Time property.
Get/Set/Reset CO_Pulse_LowTime

int32 __CFUNC DAQmxGetCOPulseLowTime(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOPulseLowTime(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCOPulseLowTime(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOPulseLowTime gets the Counter Output >> Pulse >> Time >> Low Time property.
DAQmxSetCOPulseLowTime sets the Counter Output >> Pulse >> Time >> Low Time property.
DAQmxResetCOPulseLowTime resets the Counter Output >> Pulse >> Time >> Low Time property.
Get/Set/Reset CO_Pulse_Time_InitialDelay

int32 __CFUNC DAQmxGetCOPulseTimeInitialDelay(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOPulseTimeInitialDelay(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCOPulseTimeInitialDelay(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCOPulseTimeInitialDelay gets the property.

DAQmxSetCOPulseTimeInitialDelay sets the property.

DAQmxResetCOPulseTimeInitialDelay resets the property.
Get/Set/Reset CO_Pulse_DutyCyc

int32 ___CFUNC DAQmxGetCOPulseDutyCyc(TaskHandle taskHandle, const char channel[], float64 *data);

int32 ___CFUNC DAQmxSetCOPulseDutyCyc(TaskHandle taskHandle, const char channel[], float64 data);

int32 ___CFUNC DAQmxResetCOPulseDutyCyc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOPulseDutyCyc gets the **Counter Output >> Pulse >> Frequency >> Duty Cycle** property.

DAQmxSetCOPulseDutyCyc sets the **Counter Output >> Pulse >> Frequency >> Duty Cycle** property.

DAQmxResetCOPulseDutyCyc resets the **Counter Output >> Pulse >> Frequency >> Duty Cycle** property.
Get/Set/Reset CO_Pulse_Freq_Units

int32 __CFUNC DAQmxGetCOPulseFreqUnits(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCOPulseFreqUnits(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCOPulseFreqUnits(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOPulseFreqUnits gets the Counter Output >> Pulse >> Frequency >> Units property.

DAQmxSetCOPulseFreqUnits sets the Counter Output >> Pulse >> Frequency >> Units property.

DAQmxResetCOPulseFreqUnits resets the Counter Output >> Pulse >> Frequency >> Units property.
**Get/Set/Reset CO_Pulse_Freq**

```c
int32 __CFUNC DAQmxGetCOPulseFreq(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOPulseFreq(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCOPulseFreq(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCOPulseFreq gets the `Counter Output >> Pulse >> Frequency >> Frequency` property.
DAQmxSetCOPulseFreq sets the `Counter Output >> Pulse >> Frequency >> Frequency` property.
DAQmxResetCOPulseFreq resets the `Counter Output >> Pulse >> Frequency >> Frequency` property.
Get/Set/Reset CO_Pulse_Freq_InitialDelay

int32 __CFUNC DAQmxGetCOPulseFreqInitialDelay(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOPulseFreqInitialDelay(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCOPulseFreqInitialDelay(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCOPulseFreqInitialDelay gets the `Counter Output >> Pulse >> Frequency >> Initial Delay` property.

DAQmxSetCOPulseFreqInitialDelay sets the `Counter Output >> Pulse >> Frequency >> Initial Delay` property.

DAQmxResetCOPulseFreqInitialDelay resets the `Counter Output >> Pulse >> Frequency >> Initial Delay` property.
Get/Set/Reset CO_Pulse_HighTicks

int32 __CFUNC DAQmxGetCOPulseHighTicks(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCOPulseHighTicks(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCOPulseHighTicks(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCOPulseHighTicks gets the [Counter Output >> Pulse >> Ticks >> High Ticks](#) property.

DAQmxSetCOPulseHighTicks sets the [Counter Output >> Pulse >> Ticks >> High Ticks](#) property.

DAQmxResetCOPulseHighTicks resets the [Counter Output >> Pulse >> Ticks >> High Ticks](#) property.
Get/Set/Reset CO_Pulse_LowTicks

int32 __CFUNC DAQmxGetCOPulseLowTicks(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCOPulseLowTicks(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCOPulseLowTicks(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOPulseLowTicks gets the `Counter Output >> Pulse >> Ticks >> Low Ticks` property.

DAQmxSetCOPulseLowTicks sets the `Counter Output >> Pulse >> Ticks >> Low Ticks` property.

DAQmxResetCOPulseLowTicks resets the `Counter Output >> Pulse >> Ticks >> Low Ticks` property.
Get/Set/Reset CO_Pulse_Ticks_InitialDelay

int32 __CFUNC DAQmxGetCOPulseTicksInitialDelay(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCOPulseTicksInitialDelay(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCOPulseTicksInitialDelay(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCOPulseTicksInitialDelay gets the property.

DAQmxSetCOPulseTicksInitialDelay sets the property.

DAQmxResetCOPulseTicksInitialDelay resets the property.
Get/Set/Reset CO_CtrTimebaseSrc

int32 __CFUNC DAQmxGetCOCtrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCOCtrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCOCtrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOCtrTimebaseSrc gets the Counter Output >> General Properties >> Counter Timebase >> Source property.

DAQmxSetCOCtrTimebaseSrc sets the Counter Output >> General Properties >> Counter Timebase >> Source property.

DAQmxResetCOCtrTimebaseSrc resets the Counter Output >> General Properties >> Counter Timebase >> Source property.
**Get/Set/Reset CO_CtrTimebaseRate**

```c
int32 __CFUNC DAQmxGetCOCtrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOCtrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC DAQmxResetCOCtrTimebaseRate(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCOCtrTimebaseRate gets the property.

DAQmxSetCOCtrTimebaseRate sets the property.

DAQmxResetCOCtrTimebaseRate resets the property.
Get/Set/Reset CO_CtrTimebaseActiveEdge

int32 __CFUNC DAQmxGetCOCtrTimebaseActiveEdge(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCOCtrTimebaseActiveEdge(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCOCtrTimebaseActiveEdge(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOCtrTimebaseActiveEdge gets the **Counter Output >> General Properties >> Counter Timebase >> Active Edge** property.

DAQmxSetCOCtrTimebaseActiveEdge sets the **Counter Output >> General Properties >> Counter Timebase >> Active Edge** property.

DAQmxResetCOCtrTimebaseActiveEdge resets the **Counter Output >> General Properties >> Counter Timebase >> Active Edge** property.
Get/Set/Reset CO_CtrTimebase_DigFltr_Enable

int32 __CFUNC DAQmxGetCOCtrTimebaseDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCOCtrTimebaseDigFltrEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCOCtrTimebaseDigFltrEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOCtrTimebaseDigFltrEnable gets the \texttt{Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Enable} property.

DAQmxSetCOCtrTimebaseDigFltrEnable sets the \texttt{Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Enable} property.

DAQmxResetCOCtrTimebaseDigFltrEnable resets the \texttt{Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Enable} property.
Get/Set/Reset
CO_CtrTimebase_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetCOCtrTimebaseDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOCtrTimebaseDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCOCtrTimebaseDigFltrMinPulseWidth(TaskHandle taskHandle, const char channel[]);
**Purpose**

DAQmxGetCOCtrTimebaseDigFltrMinPulseWidth gets the **Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width** property.

DAQmxSetCOCtrTimebaseDigFltrMinPulseWidth sets the **Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width** property.

DAQmxResetCOCtrTimebaseDigFltrMinPulseWidth resets the **Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Minimum Pulse Width** property.
Get/Set/Reset

CO_CtrTimebase_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetCOCtrTimebaseDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetCOCtrTimebaseDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetCOCtrTimebaseDigFltrTimebaseSrc(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOCtrTimebaseDigFltrTimebaseSrc gets the Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Source property.

DAQmxSetCOCtrTimebaseDigFltrTimebaseSrc sets the Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Source property.

DAQmxResetCOCtrTimebaseDigFltrTimebaseSrc resets the Counter Output >> General Properties >> Counter Timebase >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset

**CO_CtrTimebase_DigFltr_TimebaseRate**

```c
int32 __CFUNC DAQmxGetCOCtrlTimebaseDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 *data);

int32 __CFUNC DAQmxSetCOCtrlTimebaseDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[], float64 data);

int32 __CFUNC
    DAQmxResetCOCtrlTimebaseDigFltrTimebaseRate(TaskHandle taskHandle, const char channel[]);
```
**Purpose**

DAQmxGetCOCtrTimebaseDigFltrTimebaseRate gets the property.

DAQmxSetCOCtrTimebaseDigFltrTimebaseRate sets the property.

DAQmxResetCOCtrTimebaseDigFltrTimebaseRate resets the property.
Get/Set/Reset CO_CtrTimebase_DigSync_Enable

int32 __CFUNC DAQmxGetCOCtrTimebaseDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 *data);

int32 __CFUNC DAQmxSetCOCtrTimebaseDigSyncEnable(TaskHandle taskHandle, const char channel[], bool32 data);

int32 __CFUNC DAQmxResetCOCtrTimebaseDigSyncEnable(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOCtrTimebaseDigSyncEnable gets the Counter Output >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable property.

DAQmxSetCOCtrTimebaseDigSyncEnable sets the Counter Output >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable property.

DAQmxResetCOCtrTimebaseDigSyncEnable resets the Counter Output >> General Properties >> Counter Timebase >> Digital Synchronization >> Enable property.
Get/Set/Reset CO_Count

int32 __CFUNC DAQmxGetCOCount(TaskHandle taskHandle, const char* channel[], uint32 *data);
Purpose

DAQmxGetCOCount gets the [Counter Output >> General Properties >> More >> Count](counter_output_general_properties_more_count) property.
Get/Set/Reset CO_OutputState

int32 __CFUNC DAQmxGetCOOutputState(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetCOOutputState gets the property.
Get/Set/Reset CO_AutoIncrCnt

int32 __CFUNC DAQmxGetCOAutoIncrCnt(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCOAutoIncrCnt(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCOAutoIncrCnt(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOAutoIncrCnt gets the Counter Output >> General Properties >> More >> Auto Increment Count property.

DAQmxSetCOAutoIncrCnt sets the Counter Output >> General Properties >> More >> Auto Increment Count property.

DAQmxResetCOAutoIncrCnt resets the Counter Output >> General Properties >> More >> Auto Increment Count property.
**Get/Set/Reset CO_CtrTimebaseMasterTimebaseDiv**

```c
int32 __CFUNC DAQmxGetCOCtrlTimebaseMasterTimebaseDiv(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSetCOCtrlTimebaseMasterTimebaseDiv(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxResetCOCtrlTimebaseMasterTimebaseDiv(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCOCtrTimebaseMasterTimebaseDiv gets the property.

DAQmxSetCOCtrTimebaseMasterTimebaseDiv sets the property.

DAQmxResetCOCtrTimebaseMasterTimebaseDiv resets the property.
Get/Set/Reset CO_PulseDone

int32 __CFUNC DAQmxGetCOPulseDone(TaskHandle taskHandle, const char channel[], bool32 *data);
Purpose

DAQmxGetCOPulseDone gets the Counter Output >> General Properties >> More >> Pulse Done property.
**Get/Set/Reset CO_ConstrainedGenMode**

```
int32 __CFUNC DAQmxGetCOConstrainedGenMode(TaskHandle taskHandle, const char channel[], int32 *data);

int32 __CFUNC DAQmxSetCOConstrainedGenMode(TaskHandle taskHandle, const char channel[], int32 data);

int32 __CFUNC DAQmxResetCOConstrainedGenMode(TaskHandle taskHandle, const char channel[]);
```
Purpose

DAQmxGetCOConstrainedGenMode gets the property.

DAQmxSetCOConstrainedGenMode sets the property.

DAQmxResetCOConstrainedGenMode resets the property.
Get/Set/Reset CO_Prescaler

int32 __CFUNC DAQmxGet Copprescaler(TaskHandle taskHandle, const char channel[], uInt32 *data);

int32 __CFUNC DAQmxSet Copprescaler(TaskHandle taskHandle, const char channel[], uInt32 data);

int32 __CFUNC DAQmxReset Copprescaler(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetCOPrescaler gets the property.

DAQmxSetCOPrescaler sets the property.

DAQmxResetCOPrescaler resets the property.
Get/Set/Reset CO_RdyForNewVal

int32 __CFUNC DAQmxGetCORdyForNewVal(TaskHandle taskHandle, const char channel[], bool32 *data);
Purpose

DAQmxGetCORdyForNewVal gets the Counter Output >> General Properties >> More >> Advanced >> Ready For New Value property.
Get/Set/Reset ChanType

int32 __CFUNC DAQmxGetChanType(TaskHandle taskHandle, const char channel[], int32 *data);
Purpose

DAQmxGetChanType gets the [General Properties >> Channel Type] property.
Get/Set/Reset PhysicalChanName

int32 __CFUNC DAQmxGetPhysicalChanName(TaskHandle taskHandle, const char channel[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetPhysicalChanName(TaskHandle taskHandle, const char channel[], const char *data);
Purpose

DAQmxGetPhysicalChanName gets the General Properties >> Physical Channel Name property.
DAQmxSetPhysicalChanName sets the General Properties >> Physical Channel Name property.
Get/Set/Reset ChanDescr

int32 __CFUNC DAQmxGetChanDescr(TaskHandle taskHandle, const char channel[], char *data, uint32 bufferSize);

int32 __CFUNC DAQmxSetChanDescr(TaskHandle taskHandle, const char channel[], const char *data);

int32 __CFUNC DAQmxResetChanDescr(TaskHandle taskHandle, const char channel[]);
Purpose

DAQmxGetChanDescr gets the General Properties >> Description property.
DAQmxSetChanDescr sets the General Properties >> Description property.
DAQmxResetChanDescr resets the General Properties >> Description property.
Get/Set/Reset ChanIsGlobal

int32 __CFUNC DAQmxGetChanIsGlobal(TaskHandle taskHandle, const char channel[], bool32 *data);
Purpose

Get/Set/Reset Dev_IsSimulated

int32 __CFUNC DAQmxGetDevIsSimulated(const char device[], bool32 *data);
Purpose

DAQmxGetDevIsSimulated gets the Device Is Simulated property.
Get/Set/Reset Dev_ProductCategory

int32 __CFUNC DAQmxGetDevProductCategory(const char device[], int32 *data);
Purpose

DAQmxGetDevProductCategory gets the Identification >> Product Category property.
Get/Set/Reset Dev_ProductType

int32 __CFUNC DAQmxGetDevProductType(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevProductType gets the Identification Product Type property.
Get/Set/Reset Dev_ProductNum

int32 __CFUNC DAQmxGetDevProductNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevProductNum gets the Identification >> Product Number property.
Get/Set/Reset Dev_SerialNum

int32 __CFUNC DAQmxGetDevSerialNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevSerialNum gets the Identification Device Serial Numbers property.
Get/Set/Reset Dev_Chassis_ModuleDevNames

int32 __CFUNC DAQmxGetDevChassisModuleDevNames(const char device[], char *data, uint32 bufferSize);
Purpose

DAQmxGetDevChassisModuleDevNames gets the Chassis Module Device Names property.
Get/Set/Reset Dev_AnlgTrigSupported

int32 __CFUNC DAQmxGetDevAnlgTrigSupported(const char device[], bool32 *data);
Purpose

DAQmxGetDevAnlgTrigSupported gets the Analog Triggering Supported property.
Get/Set/Reset Dev_DigTrigSupported

int32 __CFUNC DAQmxGetDevDigTrigSupported(const char device[], bool32 *data);
Purpose

DAQmxGetDevDigTrigSupported gets the Digital Triggering Supported property.
Get/Set/Reset Dev_AI_PhysicalChans

int32 __CFUNC DAQmxGetDevAlPhysicalChans(const char device[], char *data, uint32 bufferSize);
Purpose

DAQmxGetDevAIPhysicalChans gets the \texttt{I/O Type >> Analog Input >> Physical Channels} property.
Get/Set/Reset Dev_AI_MaxSingleChanRate

int32 __CFUNC DAQmxGetDevAIMaxSingleChanRate(const char device[], float64 *data);
Purpose

DAQmxGetDevAIMaxSingleChanRate gets the I/O Type >> Analog Input >> Timing >> Maximum Single Channel Rate property.
Get/Set/Reset Dev_AI_MaxMultiChanRate

int32 __CFUNC DAQmxGetDevAIMaxMultiChanRate(const char device[], float64 *data);
Purpose

DAQmxGetDevAIMaxMultiChanRate gets the I/O Type >> Analog Input >> Timing >> Maximum Multiple Channel Rate property.
Get/Set/Reset Dev_AI_MinRate

int32 __CFUNC DAQmxGetDevAIMinRate(const char device[], float64 *data);
Purpose

DAQmxGetDevAIMinRate gets the I/O Type >> Analog Input >> Timing >> Minimum Rate property.
Get/Set/Reset
Dev_AI_SimultaneousSamplingSupported

int32 __CFUNC DAQmxGetDevAI SimultaneousSamplingSupported(const char device[], bool32 *data);
Purpose

DAQmxGetDevAISimultaneousSamplingSupported gets the I/O Type >> Analog Input >> Timing >> Simultaneous Sampling Supported property.
Get/Set/Reset Dev_AI_TrigUsage

```c
int32 __CFUNC DAQmxGetDevAITrigUsage(const char device[], int32 *data);
```
Purpose

DAQmxGetDevAITrigUsage gets the I/O Type >> Analog Input >> Trigger >> Trigger Usage property.
Get/Set/Reset Dev_AI_VoltageRngs

int32 __CFUNC DAQmxGetDevAIVoltageRngs(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAIVoltageRngs gets the I/O Type >> Analog Input >> Voltage >> Ranges property.
Get/Set/Reset Dev_AI_VoltageIntExcitDiscreteVals

int32 __CFUNC DAQmxGetDevAIVoltageIntExcitDiscreteVals(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAIVoltageIntExcitDiscreteVals gets the I/O Type >> Analog Input >> Voltage >> Internal Excitation >> Discrete Values property.
Get/Set/Reset Dev_AI_VoltageIntExcitRangeVals

int32 __CFUNC DAQmxGetDevAIVoltageIntExcitRangeVals(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAIVoltageIntExcitRangeVals gets the \textit{I/O Type >> Analog Input >> Voltage >> Internal Excitation >> Range Values} property.
Get/Set/Reset Dev_AI_CurrentRngs

int32 __CFUNC DAQmxGetDevAICurrentRngs(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAICurrentRngs gets the I/O Type >> Analog Input >> Current >> Ranges property.
Get/Set/Reset Dev_AI_CurrentIntExcitDiscreteVals

int32 __CFUNC DAQmxGetDevAICurrentIntExcitDiscreteVals(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAICurrentIntExcitDiscreteVals gets the I/O Type >> Analog Input >> Current >> Internal Excitation >> Discrete Values property.
Get/Set/Reset Dev_AI_FreqRngs

int32 __CFUNC DAQmxGetDevAIFreqRngs(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAIFreqRngs gets the **I/O Type >> Analog Input >> Frequency >> Ranges** property.
Get/Set/Reset Dev_AI_Gains

int32 __CFUNC DAQmxGetDevAI_Gains(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAI Gains gets the I/O Type >> Analog Input >> Gains property.
Get/Set/Reset Dev_AI_Couplings

int32 __CFUNC DAQmxGetDevAICouplings(const char device[], int32 *data);
Purpose

DAQmxGetDevAICouplings gets the I/O Type >> Analog Input >> Couplings property.
Get/Set/Reset

Dev_AI_LowpassCutoffFreqDiscreteVals

int32 __CFUNC DAQmxGetDevAILowpassCutoffFreqDiscreteVals(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAILowpassCutoffFreqDiscreteVals gets the I/O Type >> Analog Input >> Filter >> Analog Low Pass >> Cutoff Frequency >> Discrete Values property.
Get/Set/Reset Dev_AI_LowpassCutoffFreqRangeVals

int32 __CFUNC DAQmxGetDevAILowpassCutoffFreqRangeVals(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAILowpassCutoffFreqRangeVals gets the I/O Type >> Analog Input >> Filter >> Analog Low Pass >> Cutoff Frequency >> Range Values property.
Get/Set/Reset Dev_AO_PhysicalChans

int32 __CFUNC DAQmxGetDevAOPhysicalChans(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevAOPhysicalChans gets the `I/O Type >> Analog Output >> Physical Channels` property.
Get/Set/Reset Dev_AO_SampClkSupported

int32 __CFUNC DAQmxGetDevAOSampClkSupported(const char device[],
bool32 *data);
Purpose

DAQmxGetDevAOSampClkSupported gets the **Supported** property.
Get/Set/Reset Dev_AO_MaxRate

int32 __CFUNC DAQmxGetDevAOMaxRate(const char device[], float64 *data);
Purpose

DAQmxGetDevAOMaxRate gets the I/O Type >> Analog Output >> Timing >> Maximum Rate property.
Get/Set/Reset Dev_AO_MinRate

int32 __CFUNC DAQmxGetDevAOMinRate(const char device[], float64 *data);
Purpose

DAQmxGetDevAOMinRate gets the I/O Type >> Analog Output >> Timing >> Minimum Rate property.
Get/Set/Reset Dev_AO_TrigUsage

int32 __CFUNC DAQmxGetDevAOTrigUsage(const char device[], int32 *data);
Purpose

DAQmxGetDevAOTrigUsage gets the I/O Type >> Analog Output >> Trigger >> Trigger Usage property.
Get/Set/Reset Dev_AO_VoltageRngs

int32 __CFUNC DAQmxGetDevAOVoltageRngs(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAOVoltageRngs gets the `I/O Type >> Analog Output >> Voltage >> Ranges` property.
Get/Set/Reset Dev_AO_CurrentRngs

int32 __CFUNC DAQmxGetDevAOCurrentRngs(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAOCurrentRngs gets the `I/O Type >> Analog Output >> Current >> Ranges` property.
Get/Set/Reset Dev_AO_Gains

int32 __CFUNC DAQmxGetDevAOGains(const char device[], float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetDevAOGains gets the I/O Type >> Analog Output >> Gains property.
Get/Set/Reset Dev_DI_Lines

int32 __CFUNC DAQmxGetDevDILines(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevDILines gets the IO Type >> Digital Input >> Lines property.
Get/Set/Reset Dev_DI_Ports

int32 __CFUNC DAQmxGetDevDIPort(const char device[], char *data, uint32 bufferSize);
Purpose

DAQmxGetDevDIPorts gets the I/O Type >> Digital Input >> Ports property.
Get/Set/Reset Dev_DI_MaxRate

int32 __CFUNC DAQmxGetDevDIMaxRate(const char device[], float64 *data);
Purpose

DAQmxGetDevDIMaxRate gets the \texttt{I/O Type \rightarrow Digital Input \rightarrow Timing \rightarrow Maximum Rate} property.
Get/Set/Reset Dev_DI_TrigUsage

int32 __CFUNC DAQmxGetDevDITrigUsage(const char device[], int32 *data);
Purpose

DAQmxGetDevDITrigUsage gets the I/O Type >> Digital Input >> Trigger >> Trigger Usage property.
Get/Set/Reset Dev_DO_Lines

int32 __CFUNC DAQmxGetDevDOLines(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevDOLines gets the *I/O Type -> Digital Output -> Lines* property.
Get/Set/Reset Dev.DO_Ports

int32 __CFUNC DAQmxGetDevDOPorts(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevDOPorts gets the IO Type >> Digital Output >> Ports property.
Get/Set/Reset Dev_DO_MaxRate

```c
int32 __CFUNC DAQmxGetDevDOMaxRate(const char device[], float64 *data);
```
Purpose

DAQmxGetDevDOMaxRate gets the \textit{I/O Type >> Digital Output >> Timing >> Maximum Rate} property.
Get/Set/Reset Dev.DO_TrigUsage

int32 __CFUNC DAQmxGetDevDOTrigUsage(const char device[], int32 *data);
Purpose

DAQmxGetDevDOTrigUsage gets the I/O Type >> Digital Output >> Trigger >> Trigger Usage property.
Get/Set/Reset Dev_CI_PhysicalChans

int32 __CFUNC DAQmxGetDevCIPhysicalChans(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevCIPhysicalChans gets the I/O Type >> Counter Input >> Physical Channels property.
Get/Set/Reset Dev_CI_TrigUsage

int32 __CFUNC DAQmxGetDevCITrigUsage(const char device[], int32 *data);
Purpose

DAQmxGetDevCITrigUsage gets the `I/O Type >> Counter Input >> Trigger >> Trigger Usage` property.
Get/Set/Reset Dev_CI_SampClkSupported

int32 __CFUNC DAQmxGetDevCISampClkSupported(const char device[],
bool32 *data);
Purpose

DAQmxGetDevCISampClkSupported gets the \texttt{IO Type >> Counter Input >> Timing >> Sample Clock Supported} property.
Get/Set/Reset Dev_CI_MaxSize

int32 __CFUNC DAQmxGetDevCIMaxSize(const char device[], uInt32 *data);
Purpose

DAQmxGetDevCIMaxSize gets the [I/O Type >> Counter Input >> Maximum Size] property.
Get/Set/Reset Dev_CI_MaxTimebase

int32 __CFUNC DAQmxGetDevCIMaxTimebase(const char device[], float64 *data);
Purpose

DAQmxGetDevCIMaxTimebase gets the \textit{I/O Type >> Counter Input >> Maximum Timebase} property.
Get/Set/Reset Dev_CO_PhysicalChans

int32 __CFUNC DAQmxGetDevCOPhysicalChans(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevCOPhysicalChans gets the **I/O Type >> Counter Output >> Physical Channels** property.
Get/Set/Reset Dev_CO_TrigUsage

int32 __CFUNC DAQmxGetDevCOTrigUsage(const char device[], int32 *data);
Purpose

DAQmxGetDevCOTrigUsage gets the I/O Type >> Counter Output >> Trigger >> Trigger Usage property.
Get/Set/Reset Dev_CO_MaxSize

int32 __CFUNC DAQmxGetDevCOMaxSize(const char device[], uInt32 *data);
### Purpose

DAQmxGetDevCOMaxSize gets the `I/O Type >> Counter Output >> Maximum Size` property.
Get/Set/Reset Dev_CO_MaxTimebase

```c
int32 __CFUNC DAQmxGetDevCOMaxTimebase(const char device[], float64 *data);
```
Purpose

DAQmxGetDevCOMaxTimebase gets the \texttt{I/O Type >> Counter Output >> Maximum Timebase} property.
Get/Set/Reset Dev_BusType

int32 __CFUNC DAQmxGetDevBusType(const char device[], int32 *data);
Purpose

DAQmxGetDevBusType gets the Location >> Bus Type property.
Get/Set/Reset Dev_NumDMAChans

int32 __CFUNC DAQmxGetDevNumDMAChans(const char device[], uInt32 *data);
Purpose

DAQmxGetDevNumDMAChans gets the Bus >> Number of DMA Channels property.
Get/Set/Reset Dev_PCI_BusNum

int32 __CFUNC DAQmxGetDevPCIBusNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevPCIBusNum gets the Location >> PCI >> Bus Number property.
Get/Set/Reset Dev_PCI_DevNum

int32 __CFUNC DAQmxGetDevPCIDevNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevPCIDevNum gets the Location >> PCI >> Device Number property.
Get/Set/Reset Dev_PXI_ChassisNum

int32 __CFUNC DAQmxGetDevPXIClassNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevPXIClassisNum gets the Location >> PXI >> Chassis Number property.
Get/Set/Reset Dev_PXI_SlotNum

int32 __CFUNC DAQmxGetDevPXISlotNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevPXISlotNum gets the [Location >> PXI >> Slot Number] property.
Get/Set/Reset Dev_CompactDAQ_ChassisDevName

int32 __CFUNC DAQmxGetDevCompactDAQChassisDevName(const char *device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevCompactDAQChassisDevName gets the Location >> CompactDAQ >> Chassis Device Name property.
Get/Set/Reset Dev_CompactDAQ_SlotNum

int32 __CFUNC DAQmxGetDevCompactDAQSlotNum(const char device[], uInt32 *data);
Purpose

DAQmxGetDevCompactDAQSlotNum gets the Location >> CompactDAQ >> Slot Number property.
Get/Set/Reset Dev_Terminals

int32 __CFUNC DAQmxGetDevTerminals(const char device[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetDevTerminals gets the `Terminals` property.
Get/Set/Reset Exported_AICnvClk_OutputTerm

int32 __CFUNC DAQmxGetExportedAIConvClkOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedAIConvClkOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedAIConvClkOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedAIConvClkOutputTerm gets the [Clocks >> AI Convert Clock >> Output Terminal](#) property.

DAQmxSetExportedAIConvClkOutputTerm sets the [Clocks >> AI Convert Clock >> Output Terminal](#) property.

DAQmxResetExportedAIConvClkOutputTerm resets the [Clocks >> AI Convert Clock >> Output Terminal](#) property.
Get/Set/Reset Exported_AIConvClk_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedAIConvClkPulsePolarity(TaskHandle taskHandle, int32 *data);
Purpose

DAQmxGetExportedAIConvClkPulsePolarity gets the Clocks >> AI Convert Clock >> Pulse >> Polarity property.
Get/Set/Reset Exported_10MHzRefClk_OutputTerm

int32 __CFUNC DAQmxGetExported10MHzRefClkOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExported10MHzRefClkOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExported10MHzRefClkOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExported10MHzRefClkOutputTerm gets the Clocks >> 10MHz Reference Clock >> Output Terminal property.

DAQmxSetExported10MHzRefClkOutputTerm sets the Clocks >> 10MHz Reference Clock >> Output Terminal property.

DAQmxResetExported10MHzRefClkOutputTerm resets the Clocks >> 10MHz Reference Clock >> Output Terminal property.
Get/Set/Reset
Exported_20MHzTimebase_OutputTerm

int32 __CFUNC DAQmxGetExported20MHzTimebaseOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExported20MHzTimebaseOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExported20MHzTimebaseOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExported20MHzTimebaseOutputTerm gets the Clocks >> 20MHz Timebase >> Output Terminal property.

DAQmxSetExported20MHzTimebaseOutputTerm sets the Clocks >> 20MHz Timebase >> Output Terminal property.

DAQmxResetExported20MHzTimebaseOutputTerm resets the Clocks >> 20MHz Timebase >> Output Terminal property.
Get/Set/Reset Exported_SampClk_OutputBehavior

int32 __CFUNC DAQmxGetExportedSampClkOutputBehavior(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedSampClkOutputBehavior(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedSampClkOutputBehavior(TaskHandle taskHandle);
Purpose

DAQmxGetExportedSampClkOutputBehavior gets the Clocks >> Sample Clock >> Output Behavior property.

DAQmxSetExportedSampClkOutputBehavior sets the Clocks >> Sample Clock >> Output Behavior property.

DAQmxResetExportedSampClkOutputBehavior resets the Clocks >> Sample Clock >> Output Behavior property.
Get/Set/Reset Exported_SampClk_OutputTerm

int32 __CFUNC DAQmxGetExportedSampClkOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedSampClkOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedSampClkOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedSampClkOutputTerm gets the property.

DAQmxSetExportedSampClkOutputTerm sets the property.

DAQmxResetExportedSampClkOutputTerm resets the property.
Get/Set/Reset Exported_SampClk_DelayOffset

int32 __CFUNC DAQmxGetExportedSampClkDelayOffset(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetExportedSampClkDelayOffset(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetExportedSampClkDelayOffset(TaskHandle taskHandle);
Purpose

DAQmxGetExportedSampClkDelayOffset gets the Clocks >> Sample Clock >> Delay Offset property.

DAQmxSetExportedSampClkDelayOffset sets the Clocks >> Sample Clock >> Delay Offset property.

DAQmxResetExportedSampClkDelayOffset resets the Clocks >> Sample Clock >> Delay Offset property.
Get/Set/Reset Exported_SampClk_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedSampClkPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedSampClkPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedSampClkPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGetExportedSampClkPulsePolarity gets the property.

DAQmxSetExportedSampClkPulsePolarity sets the property.

DAQmxResetExportedSampClkPulsePolarity resets the property.
**Get/Set/Reset**

**Exported_SampClkTimebase_OutputTerm**

```c
int32 __CFUNC
    DAQmxGetExportedSampClkTimebaseOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetExportedSampClkTimebaseOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedSampClkTimebaseOutputTerm(TaskHandle taskHandle);
```
Purpose

DAQmxGetExportedSampClkTimebaseOutputTerm gets the Clocks >> Sample Clock Timebase Output Terminal property.

DAQmxSetExportedSampClkTimebaseOutputTerm sets the Clocks >> Sample Clock Timebase Output Terminal property.

DAQmxResetExportedSampClkTimebaseOutputTerm resets the Clocks >> Sample Clock Timebase Output Terminal property.
Get/Set/Reset
Exported_DividedSampClkTimebase_OutputTerm

int32 __CFUNC
    DAQmxGetExportedDividedSampClkTimebaseOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetExportedDividedSampClkTimebaseOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedDividedSampClkTimebaseOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedDividedSampClkTimebaseOutputTerm gets the [Clocks >> Divided Sample Clock Timebase >> Output Terminal](#) property.

DAQmxSetExportedDividedSampClkTimebaseOutputTerm sets the [Clocks >> Divided Sample Clock Timebase >> Output Terminal](#) property.

DAQmxResetExportedDividedSampClkTimebaseOutputTerm resets the [Clocks >> Divided Sample Clock Timebase >> Output Terminal](#) property.
Get/Set/Reset Exported_AdvTrig_OutputTerm

int32 __CFUNC DAQmxGetExportedAdvTrigOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedAdvTrigOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedAdvTrigOutputTerm(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvTrigOutputTerm gets the Triggers >> Advance Trigger >> Output Terminal property.

DAQmxSetExportedAdvTrigOutputTerm sets the Triggers >> Advance Trigger >> Output Terminal property.

DAQmxResetExportedAdvTrigOutputTerm resets the Triggers >> Advance Trigger >> Output Terminal property.
Get/Set/Reset Exported_AdvTrig_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedAdvTrigPulsePolarity(TaskHandle taskHandle, int32 *data);
Purpose

DAQmxGetExportedAdvTrigPulsePolarity gets the `Triggers >> Advance Trigger >> Pulse >> Polarity` property.
Get/Set/Reset Exported_AdvTrig_Pulse_WidthUnits

int32 __CFUNC DAQmxGetExportedAdvTrigPulseWidthUnits(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedAdvTrigPulseWidthUnits(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedAdvTrigPulseWidthUnits(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvTrigPulseWidthUnits gets the Triggers >> Advance Trigger >> Pulse >> Width Units property.

DAQmxSetExportedAdvTrigPulseWidthUnits sets the Triggers >> Advance Trigger >> Pulse >> Width Units property.

DAQmxResetExportedAdvTrigPulseWidthUnits resets the Triggers >> Advance Trigger >> Pulse >> Width Units property.
Get/Set/Reset Exported_AdvTrig_Pulse_Width

int32 __CFUNC DAQmxGetExportedAdvTrigPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetExportedAdvTrigPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetExportedAdvTrigPulseWidth(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvTrigPulseWidth gets the \textit{Triggers \gg Advance Trigger \gg Pulse \gg Width Value} property.

DAQmxSetExportedAdvTrigPulseWidth sets the \textit{Triggers \gg Advance Trigger \gg Pulse \gg Width Value} property.

DAQmxResetExportedAdvTrigPulseWidth resets the \textit{Triggers \gg Advance Trigger \gg Pulse \gg Width Value} property.
Get/Set/Reset Exported_PauseTrig_OutputTerm

int32 __CFUNC DAQmxGetExportedPauseTrigOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedPauseTrigOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedPauseTrigOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedPauseTrigOutputTerm gets the `Triggers >> Pause Trigger >> Output Terminal` property.

DAQmxSetExportedPauseTrigOutputTerm sets the `Triggers >> Pause Trigger >> Output Terminal` property.

DAQmxResetExportedPauseTrigOutputTerm resets the `Triggers >> Pause Trigger >> Output Terminal` property.
Get/Set/Reset Exported_PauseTrig_Lvl_ActiveLvl

int32 __CFUNC DAQmxGetExportedPauseTrigLvlActiveLvl(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedPauseTrigLvlActiveLvl(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedPauseTrigLvlActiveLvl(TaskHandle taskHandle);
Purpose

DAQmxGetExportedPauseTrigLvlActiveLvl gets the Triggers >> Pause Trigger >> Level >> Active Level property.

DAQmxSetExportedPauseTrigLvlActiveLvl sets the Triggers >> Pause Trigger >> Level >> Active Level property.

DAQmxResetExportedPauseTrigLvlActiveLvl resets the Triggers >> Pause Trigger >> Level >> Active Level property.
Get/Set/Reset Exported_RefTrig_OutputTerm

int32 __CFUNC DAQmxGetExportedRefTrigOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedRefTrigOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedRefTrigOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedRefTrigOutputTerm gets the Triggers >> Reference Trigger >> Output Terminal property.

DAQmxSetExportedRefTrigOutputTerm sets the Triggers >> Reference Trigger >> Output Terminal property.

DAQmxResetExportedRefTrigOutputTerm resets the Triggers >> Reference Trigger >> Output Terminal property.
Get/Set/Reset Exported_RefTrig_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedRefTrigPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedRefTrigPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedRefTrigPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGetExportedRefTrigPulsePolarity gets the Triggers >> Reference Trigger >> Pulse >> Polarity property.

DAQmxSetExportedRefTrigPulsePolarity sets the Triggers >> Reference Trigger >> Pulse >> Polarity property.

DAQmxResetExportedRefTrigPulsePolarity resets the Triggers >> Reference Trigger >> Pulse >> Polarity property.
**Get/Set/Reset Exported_StartTrig_OutputTerm**

```c
int32 __CFUNC DAQmxGetExportedStartTrigOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedStartTrigOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedStartTrigOutputTerm(TaskHandle taskHandle);
```
**Purpose**

DAQmxGetExportedStartTrigOutputTerm gets the Triggers >> Start Trigger >> Output Terminal property.

DAQmxSetExportedStartTrigOutputTerm sets the Triggers >> Start Trigger >> Output Terminal property.

DAQmxResetExportedStartTrigOutputTerm resets the Triggers >> Start Trigger >> Output Terminal property.
Get/Set/Reset Exported_StartTrig_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedStartTrigPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedStartTrigPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedStartTrigPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGetExportedStartTrigPulsePolarity gets the Triggers >> Start Trigger >> Pulse >> Polarity property.

DAQmxSetExportedStartTrigPulsePolarity sets the Triggers >> Start Trigger >> Pulse >> Polarity property.

DAQmxResetExportedStartTrigPulsePolarity resets the Triggers >> Start Trigger >> Pulse >> Polarity property.
Get/Set/Reset

Exported_AdvCmpltEvent_OutputTerm

int32 __CFUNC DAQmxGetExportedAdvCmpltEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedAdvCmpltEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedAdvCmpltEventOutputTerm(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvCmpltEventOutputTerm gets the Events >> Advance Complete Event >> Output Terminal property.

DAQmxSetExportedAdvCmpltEventOutputTerm sets the Events >> Advance Complete Event >> Output Terminal property.

DAQmxResetExportedAdvCmpltEventOutputTerm resets the Events >> Advance Complete Event >> Output Terminal property.
Get/Set/Reset Exported_AdvCmpltEvent_Delay

int32 __CFUNC DAQmxGetExportedAdvCmpltEventDelay(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetExportedAdvCmpltEventDelay(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetExportedAdvCmpltEventDelay(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvCmpltEventDelay gets the Events >> Advance Complete Event >> Delay Value property.

DAQmxSetExportedAdvCmpltEventDelay sets the Events >> Advance Complete Event >> Delay Value property.

DAQmxResetExportedAdvCmpltEventDelay resets the Events >> Advance Complete Event >> Delay Value property.
Get/Set/Reset
Exported_AdvCmpltEvent_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedAdvCmpltEventPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedAdvCmpltEventPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedAdvCmpltEventPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvCmpltEventPulsePolarity gets the Events >> Advance Complete Event >> Pulse >> Polarity property.

DAQmxSetExportedAdvCmpltEventPulsePolarity sets the Events >> Advance Complete Event >> Pulse >> Polarity property.

DAQmxResetExportedAdvCmpltEventPulsePolarity resets the Events >> Advance Complete Event >> Pulse >> Polarity property.
Get/Set/Reset

Exported_AdvCmpltEvent_Pulse_Width

int32 __CFUNC DAQmxGetExportedAdvCmpltEventPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetExportedAdvCmpltEventPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetExportedAdvCmpltEventPulseWidth(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAdvCmpltEventPulseWidth gets the Events >> Advance Complete Event >> Pulse >> Width Value property.

DAQmxSetExportedAdvCmpltEventPulseWidth sets the Events >> Advance Complete Event >> Pulse >> Width Value property.

DAQmxResetExportedAdvCmpltEventPulseWidth resets the Events >> Advance Complete Event >> Pulse >> Width Value property.
Get/Set/Reset
Exported_AIHoldCmpltEvent_OutputTerm

int32 __CFUNC
   DAQmxGetExportedAIHoldCmpltEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
   DAQmxSetExportedAIHoldCmpltEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
   DAQmxResetExportedAIHoldCmpltEventOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedAIHoldCmpltEventOutputTerm gets the **Events >> AI Hold Complete Event >> Output Terminal** property.

DAQmxSetExportedAIHoldCmpltEventOutputTerm sets the **Events >> AI Hold Complete Event >> Output Terminal** property.

DAQmxResetExportedAIHoldCmpltEventOutputTerm resets the **Events >> AI Hold Complete Event >> Output Terminal** property.
Get/Set/Reset
Exported_AIHoldCmpltEvent_PulsePolarity

int32 __CFUNC
    DAQmxGetExportedAIHoldCmpltEventPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetExportedAIHoldCmpltEventPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedAIHoldCmpltEventPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGetExportedAIHoldCmpltEventPulsePolarity gets the Events >> AI Hold Complete property.

DAQmxSetExportedAIHoldCmpltEventPulsePolarity sets the Events >> AI Hold Complete property.

DAQmxResetExportedAIHoldCmpltEventPulsePolarity resets the Events >> AI Hold Complete property.
Get/Set/Reset
EXPORTED_CHANGE_DETECT_EVENT_OUTPUTTERM

int32 __CFUNC
    DAQmxGetExportedChangeDetectEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetExportedChangeDetectEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedChangeDetectEventOutputTerm(TaskHandle taskHandle);
Purpose

**Purpose**

DAQmxGetExportedChangeDetectEventOutputTerm gets the Events >> Change Detection Event >> Output Terminal property.

DAQmxSetExportedChangeDetectEventOutputTerm sets the Events >> Change Detection Event >> Output Terminal property.

DAQmxResetExportedChangeDetectEventOutputTerm resets the Events >> Change Detection Event >> Output Terminal property.
Get/Set/Reset
Exported_ChangeDetectEvent_Pulse_Polarity

int32 __CFUNC
    DAQmxGetExportedChangeDetectEventPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetExportedChangeDetectEventPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedChangeDetectEventPulsePolarity(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedChangeDetectEventPulsePolarity gets the Events >> Change Detection Event >> Pulse >> Polarity property.

DAQmxSetExportedChangeDetectEventPulsePolarity sets the Events >> Change Detection Event >> Pulse >> Polarity property.

DAQmxResetExportedChangeDetectEventPulsePolarity resets the Events >> Change Detection Event >> Pulse >> Polarity property.
Get/Set/Reset Exported_CtrOutEvent_OutputTerm

int32 __CFUNC DAQmxGetExportedCtrOutEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedCtrOutEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedCtrOutEventOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedCtrOutEventOutputTerm gets the Events >> Counter Output Event >> Output Terminal property.

DAQmxSetExportedCtrOutEventOutputTerm sets the Events >> Counter Output Event >> Output Terminal property.

DAQmxResetExportedCtrOutEventOutputTerm resets the Events >> Counter Output Event >> Output Terminal property.
Get/Set/Reset
Exported_CtrOutEvent_OutputBehavior

int32 __CFUNC DAQmxGetExportedCtrOutEventOutputBehavior(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedCtrOutEventOutputBehavior(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedCtrOutEventOutputBehavior(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedCtrOutEventOutputBehavior gets the `Events >> Counter Output Event >> Output Behavior` property.

DAQmxSetExportedCtrOutEventOutputBehavior sets the `Events >> Counter Output Event >> Output Behavior` property.

DAQmxResetExportedCtrOutEventOutputBehavior resets the `Events >> Counter Output Event >> Output Behavior` property.
Get/Set/Reset Exported_CtrOutEvent_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedCtrOutEventPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedCtrOutEventPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedCtrOutEventPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGetExportedCtrOutEventPulsePolarity gets the Events >> Counter Output Event >> Pulse Polarity property.

DAQmxSetExportedCtrOutEventPulsePolarity sets the Events >> Counter Output Event >> Pulse Polarity property.

DAQmxResetExportedCtrOutEventPulsePolarity resets the Events >> Counter Output Event >> Pulse Polarity property.
Get/Set/Reset
Exported_CtrOutEvent_Toggle_IdleState

int32 __CFUNC DAQmxGetExportedCtrOutEventToggleIdleState(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedCtrOutEventToggleIdleState(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedCtrOutEventToggleIdleState(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedCtrOutEventToggleIdleState gets the `Events >> Counter Output Event >> Toggle >> Idle State` property.

DAQmxSetExportedCtrOutEventToggleIdleState sets the `Events >> Counter Output Event >> Toggle >> Idle State` property.

DAQmxResetExportedCtrOutEventToggleIdleState resets the `Events >> Counter Output Event >> Toggle >> Idle State` property.
Get/Set/Reset Exported_HshkEvent_OutputTerm

int32 __CFUNC DAQmxGetExportedHshkEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedHshkEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetExportedHshkEventOutputTerm(TaskHandle taskHandle);
Purpose

DAQmxGetExportedHshkEventOutputTerm gets the Events >> Handshake Event >> Output Terminal property.

DAQmxSetExportedHshkEventOutputTerm sets the Events >> Handshake Event >> Output Terminal property.

DAQmxResetExportedHshkEventOutputTerm resets the Events >> Handshake Event >> Output Terminal property.
Get/Set/Reset Exported_HshkEvent_OutputBehavior

int32 __CFUNC DAQmxGetExportedHshkEventOutputBehavior(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedHshkEventOutputBehavior(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedHshkEventOutputBehavior(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedHshkEventOutputBehavior gets the **Events >> Handshake Event >> Output Behavior** property.

DAQmxSetExportedHshkEventOutputBehavior sets the **Events >> Handshake Event >> Output Behavior** property.

DAQmxResetExportedHshkEventOutputBehavior resets the **Events >> Handshake Event >> Output Behavior** property.
Get/Set/Reset Exported_HshkEvent_Delay

int32 __CFUNC DAQmxGetExportedHshkEventDelay(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetExportedHshkEventDelay(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetExportedHshkEventDelay(TaskHandle taskHandle);
Purpose

DAQmxGetExportedHshkEventDelay gets the **Events >> Handshake Event >> Delay Value** property.

DAQmxSetExportedHshkEventDelay sets the **Events >> Handshake Event >> Delay Value** property.

DAQmxResetExportedHshkEventDelay resets the **Events >> Handshake Event >> Delay Value** property.
Get/Set/Reset

Exported_HshkEvent_Interlocked_AssertedLvl

int32 __CFUNC
    DAQmxGetExportedHshkEventInterlockedAssertedLvl(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetExportedHshkEventInterlockedAssertedLvl(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedHshkEventInterlockedAssertedLvl(TaskHandle taskHandle);
Purpose

DAQmxGetExportedHshkEventInterlockedAssertedLvl gets the Events >> Handshake Event >> Interlocked >> Asserted Level property.

DAQmxSetExportedHshkEventInterlockedAssertedLvl sets the Events >> Handshake Event >> Interlocked >> Asserted Level property.

DAQmxResetExportedHshkEventInterlockedAssertedLvl resets the Events >> Handshake Event >> Interlocked >> Asserted Level property.
Get/Set/Reset
Exported_HshkEvent_Interlocked_AssertOnStart

int32 __CFUNC
   DAQmxGetExportedHshkEventInterlockedAssertOnStart(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC
   DAQmxSetExportedHshkEventInterlockedAssertOnStart(TaskHandle taskHandle, bool32 data);

int32 __CFUNC
   DAQmxResetExportedHshkEventInterlockedAssertOnStart(TaskHandle taskHandle);
Purpose

DAQmxGetExportedHshkEventInterlockedAssertOnStart gets the Events >> Handshake Event >> Interlocked >> Assert on Start property.

DAQmxSetExportedHshkEventInterlockedAssertOnStart sets the Events >> Handshake Event >> Interlocked >> Assert on Start property.

DAQmxResetExportedHshkEventInterlockedAssertOnStart resets the Events >> Handshake Event >> Interlocked >> Assert on Start property.
Get/Set/Reset
Exported_HshkEvent_Interlocked_DeassertDelay

int32 __CFUNC
    DAQmxGetExportedHshkEventInterlockedDeassertDelay(TaskHandle taskHandle, float64 *data);

int32 __CFUNC
    DAQmxSetExportedHshkEventInterlockedDeassertDelay(TaskHandle taskHandle, float64 data);

int32 __CFUNC
    DAQmxResetExportedHshkEventInterlockedDeassertDelay(TaskHandle taskHandle);
Purpose

DAQmxGetExportedHshkEventInterlockedDeassertDelay gets the Events >> Handshake Event >> Interlocked >> Deassert Delay Value property.

DAQmxSetExportedHshkEventInterlockedDeassertDelay sets the Events >> Handshake Event >> Interlocked >> Deassert Delay Value property.

DAQmxResetExportedHshkEventInterlockedDeassertDelay resets the Events >> Handshake Event >> Interlocked >> Deassert Delay Value property.
Get/Set/Reset Exported_HshkEvent_Pulse_Polarity

int32 __CFUNC DAQmxGetExportedHshkEventPulsePolarity(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetExportedHshkEventPulsePolarity(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetExportedHshkEventPulsePolarity(TaskHandle taskHandle);
Purpose

DAQmxGet exported Hshk Event Pulse Polarity gets the Polarity property.

DAQmxSet exported Hshk Event Pulse Polarity sets the Polarity property.

DAQmxReset exported Hshk Event Pulse Polarity resets the Polarity property.
Get/Set/Reset Exported_HshkEvent_Pulse_Width

int32 __CFUNC DAQmxGetExportedHshkEventPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetExportedHshkEventPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetExportedHshkEventPulseWidth(TaskHandle taskHandle);
Purpose

DAQmxGetExportedHshkEventPulseWidth gets the Events >> Handshake Event >> Pulse >> Width Value property.

DAQmxSetExportedHshkEventPulseWidth sets the Events >> Handshake Event >> Pulse >> Width Value property.

DAQmxResetExportedHshkEventPulseWidth resets the Events >> Handshake Event >> Pulse >> Width Value property.
**Get/Set/Reset**

**Exported_RdyForXferEvent_OutputTerm**

```c
int32 __CFUNC
    DAQmxGetExportedRdyForXferEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedRdyForXferEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedRdyForXferEventOutputTerm(TaskHandle taskHandle);
```
Purpose

DAQmxGetExportedRdyForXferEventOutputTerm gets the Events >> Ready For Transfer Event Output Terminal property.

DAQmxSetExportedRdyForXferEventOutputTerm sets the Events >> Ready For Transfer Event Output Terminal property.

DAQmxResetExportedRdyForXferEventOutputTerm resets the Events >> Ready For Transfer Event Output Terminal property.
Get/Set/Reset
Exported_RdyForXferEvent_Lvl_ActiveLvl

int32 __CFUNC
    DAQmxGetExportedRdyForXferEventLvlActiveLvl(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetExportedRdyForXferEventLvlActiveLvl(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedRdyForXferEventLvlActiveLvl(TaskHandle taskHandle);
Purpose

DAQmxGetExportedRdyForXferEventLvlActiveLvl gets the Events >> Ready For Transfer Event >> Level >> Active Level property.

DAQmxSetExportedRdyForXferEventLvlActiveLvl sets the Events >> Ready For Transfer Event >> Level >> Active Level property.

DAQmxResetExportedRdyForXferEventLvlActiveLvl resets the Events >> Ready For Transfer Event >> Level >> Active Level property.
Get/Set/Reset
Exported_RdyForXferEvent_DeassertCond

int32 __CFUNC
    DAQmxGetExportedRdyForXferEventDeassertCond(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetExportedRdyForXferEventDeassertCond(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedRdyForXferEventDeassertCond(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedRdyForXferEventDeassertCond gets the Events >> Ready For Transfer Event >> Deassert Condition property.

DAQmxSetExportedRdyForXferEventDeassertCond sets the Events >> Ready For Transfer Event >> Deassert Condition property.

DAQmxResetExportedRdyForXferEventDeassertCond resets the Events >> Ready For Transfer Event >> Deassert Condition property.
Get/Set/Reset
Exported_RdyForXferEvent_DeassertCondCustomTh

int32 __CFUNC
    DAQmxGetExportedRdyForXferEventDeassertCondCustomThreshold(taskHandle, uInt32 *data);

int32 __CFUNC
    DAQmxSetExportedRdyForXferEventDeassertCondCustomThreshold(taskHandle, uInt32 data);

int32 __CFUNC
    DAQmxResetExportedRdyForXferEventDeassertCondCustomThreshold(taskHandle);
Purpose

DAQmxGetExportedRdyForXferEventDeassertCondCustomThreshold gets the property.

DAQmxSetExportedRdyForXferEventDeassertCondCustomThreshold sets the property.

DAQmxResetExportedRdyForXferEventDeassertCondCustomThreshold resets the property.
Get/Set/Reset

Exported_DataActiveEvent_OutputTerm

int32 __CFUNC DAQmxGetExportedDataActiveEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedDataActiveEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedDataActiveEventOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedDataActiveEventOutputTerm gets the **Events >> Data Active Event >> Output Terminal** property.

DAQmxSetExportedDataActiveEventOutputTerm sets the **Events >> Data Active Event >> Output Terminal** property.

DAQmxResetExportedDataActiveEventOutputTerm resets the **Events >> Data Active Event >> Output Terminal** property.
Get/Set/Reset
Exported_DataActiveEvent_Lvl_ActiveLvl

int32 __CFUNC
    DAQmxGetExportedDataActiveEventLvlActiveLvl(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetExportedDataActiveEventLvlActiveLvl(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetExportedDataActiveEventLvlActiveLvl(TaskHandle taskHandle);
Purpose

DAQmxGetExportedDataActiveEventLvlActiveLvl gets the Events >> Data Active Event >> Level >> Active Level property.

DAQmxSetExportedDataActiveEventLvlActiveLvl sets the Events >> Data Active Event >> Level >> Active Level property.

DAQmxResetExportedDataActiveEventLvlActiveLvl resets the Events >> Data Active Event >> Level >> Active Level property.
Get/Set/Reset

Exported_RdyForStartEvent_OutputTerm

int32 __CFUNC
    DAQmxGetExportedRdyForStartEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetExportedRdyForStartEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedRdyForStartEventOutputTerm(TaskHandle taskHandle);
Purpose

DAQmxGetExportedRdyForStartEventOutputTerm gets the `Events >> Ready For Start Event >> Output Terminal` property.

DAQmxSetExportedRdyForStartEventOutputTerm sets the `Events >> Ready For Start Event >> Output Terminal` property.

DAQmxResetExportedRdyForStartEventOutputTerm resets the `Events >> Ready For Start Event >> Output Terminal` property.
Get/Set/Reset
Exported_RdyForStartEvent_Lvl_ActiveLvl

int32 __CFUNC
   DAQmxGetExportedRdyForStartEventLvlActiveLvl(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
   DAQmxSetExportedRdyForStartEventLvlActiveLvl(TaskHandle taskHandle, int32 data);

int32 __CFUNC
   DAQmxResetExportedRdyForStartEventLvlActiveLvl(TaskHandle taskHandle);
### Purpose

DAQmxGetExportedRdyForStartEventLvlActiveLvl gets the \texttt{Events >> Ready For Start Event >> Level >> Active Level} property.

DAQmxSetExportedRdyForStartEventLvlActiveLvl sets the \texttt{Events >> Ready For Start Event >> Level >> Active Level} property.

DAQmxResetExportedRdyForStartEventLvlActiveLvl resets the \texttt{Events >> Ready For Start Event >> Level >> Active Level} property.
Get/Set/Reset
Exported_SyncPulseEvent_OutputTerm

int32 __CFUNC DAQmxGetExportedSyncPulseEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetExportedSyncPulseEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedSyncPulseEventOutputTerm(TaskHandle taskHandle);
Purpose

DAQmxGetExportedSyncPulseEventOutputTerm gets the Events >> Synchronization Pulse Event >> Output Terminal property.

DAQmxSetExportedSyncPulseEventOutputTerm sets the Events >> Synchronization Pulse Event >> Output Terminal property.

DAQmxResetExportedSyncPulseEventOutputTerm resets the Events >> Synchronization Pulse Event >> Output Terminal property.
Get/Set/Reset
Exported_WatchdogExpiredEvent_OutputTerm

int32 __CFUNC
    DAQmxGetExportedWatchdogExpiredEventOutputTerm(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetExportedWatchdogExpiredEventOutputTerm(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetExportedWatchdogExpiredEventOutputTerm(TaskHandle taskHandle);
**Purpose**

DAQmxGetExportedWatchdogExpiredEventOutputTerm gets the [Events >> Watchdog Timer Expired Event >> Output Terminal](#) property.

DAQmxSetExportedWatchdogExpiredEventOutputTerm sets the [Events >> Watchdog Timer Expired Event >> Output Terminal](#) property.

DAQmxResetExportedWatchdogExpiredEventOutputTerm resets the [Events >> Watchdog Timer Expired Event >> Output Terminal](#) property.
Get/Set/Reset PersistedChan_Author

int32 __CFUNC DAQmxGetPersistedChanAuthor(const char channel[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetPersistedChanAuthor gets the Author property.
Get/Set/Reset PersistedChan_AllowInteractiveEditing

int32 __CFUNC DAQmxGetPersistedChanAllowInteractiveEditing(const char channel[], bool32 *data);
Purpose

DAQmxGetPersistedChanAllowInteractiveEditing gets the Allow Interactive Editing? property.
Get/Set/Reset
PersistedChan_AllowInteractiveDeletion

int32 __CFUNC DAQmxGetPersistedChanAllowInteractiveDeletion(const char channel[], bool32 *data);
Purpose

DAQmxGetPersistedChanAllowInteractiveDeletion gets the Allow Interactive Deletion property.
Get/Set/Reset PersistedScale_Author

int32 __CFUNC DAQmxGetPersistedScaleAuthor(const char scaleName[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetPersistedScaleAuthor gets the Author property.
Get/Set/Reset PersistedScale_AllowInteractiveEditing

int32 __CFUNC DAQmxGetPersistedScaleAllowInteractiveEditing(const char * scaleName[], bool32 *data);
Purpose

DAQmxGetPersistedScaleAllowInteractiveEditing gets the Allow Interactive Editing property.
Get/Set/Reset
PersistedScale_AllowInteractiveDeletion

int32 __CFUNC DAQmxGetPersistedScaleAllowInteractiveDeletion(const char scaleName[], bool32 *data);
Purpose

DAQmxGetPersistedScaleAllowInteractiveDeletion gets the Allow Interactive Deletion property.
Get/Set/Reset PersistedTask_Author

int32 __CFUNC DAQmxGetPersistedTaskAuthor(const char taskName[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetPersistedTaskAuthor gets the Author property.
Get/Set/Reset PersistedTask_AllowInteractiveEditing

int32 __CFUNC DAQmxGetPersistedTaskAllowInteractiveEditing(const char *taskName[], bool32 *data);
Purpose

DAQmxGetPersistedTaskAllowInteractiveEditing gets the Allow Interactive Editing? property.
Get/Set/Reset PersistedTask_AllowInteractiveDeletion

int32 __CFUNC DAQmxGetPersistedTaskAllowInteractiveDeletion(const char taskName[], bool32 *data);
Purpose

DAQmxGetPersistedTaskAllowInteractiveDeletion gets the Allow Interactive Deletion property.
Get/Set/Reset PhysicalChan_AI_TermCfgs

int32 __CFUNC DAQmxGetPhysicalChanAITermCfgs(const char *physicalChannel[], int32 *data);
Purpose

DAQmxGetPhysicalChanAITermCfgs gets the Analog Input >> Input Configuration >> Terminal Configurations property.
Get/Set/Reset PhysicalChan_AO_TermCfgs

int32 __CFUNC DAQmxGetPhysicalChanAOTermCfgs(const char *physicalChannel[], int32 *data);
Purpose

DAQmxGetPhysicalChanAOTermCfgs gets the Analog Output >> Output Configuration >> Terminal Configurations property.
Get/Set/Reset
PhysicalChan_AO_ManualControlEnable

int32 __CFUNC DAQmxGetPhysicalChanAOManualControlEnable(const char physicalChannel[], bool32 *data);

int32 __CFUNC DAQmxSetPhysicalChanAOManualControlEnable(const char physicalChannel[], bool32 data);

int32 __CFUNC DAQmxResetPhysicalChanAOManualControlEnable(const char physicalChannel[]);
**Purpose**

DAQmxGetPhysicalChanAOManualControlEnable gets the **property**.

DAQmxSetPhysicalChanAOManualControlEnable sets the **property**.

DAQmxResetPhysicalChanAOManualControlEnable resets the **property**.
Get/Set/Reset
PhysicalChan_AO_ManualControlAmplitude

int32 __CFUNC DAQmxGetPhysicalChanAOManualControlAmplitude(const char physicalChannel[], float64 *data);
Purpose

Get/Set/Reset PhysicalChan_AO_ManualControlFreq

int32 __CFUNC DAQmxGetPhysicalChanAOManualControlFreq(const char *physicalChannel[], float64 *data);
Purpose

DAQmxGetPhysicalChanAOManualControlFreq gets the Analog Output >> Advanced >> Manual Control >> Frequency property.
Get/Set/Reset PhysicalChan_DI_PortWidth

int32 __CFUNC DAQmxGetPhysicalChanDIPortWidth(const char physicalChannel[], uInt32 *data);
Purpose

DAQmxGetPhysicalChanDIPortWidth gets the Digital Input >> Port Width property.
Get/Set/Reset PhysicalChan_DI_SampClkSupported

int32 __CFUNC DAQmxGetPhysicalChanDISampClkSupported(const char physicalChannel[], bool32 *data);
Purpose

DAQmxGetPhysicalChanDISampClkSupported gets the \texttt{Digital Input >> Timing >> Sample Clock Supported} property.
Get/Set/Reset

**PhysicalChan_DI_ChangeDetectSupported**

```c
int32 __CFUNC DAQmxGetPhysicalChanDIChangeDetectSupported(const char physicalChannel[], bool32 *data);
```
Purpose

Get/Set/Reset PhysicalChan_DO_PortWidth

int32 __CFUNC DAQmxGetPhysicalChanDOPortWidth(const char physicalChannel[], uInt32 *data);
Purpose

DAQmxGetPhysicalChanDOPortWidth gets the Digital Output >> Port Width property.
Get/Set/Reset PhysicalChan.DO_SampClkSupported

int32 __CFUNC DAQmxGetPhysicalChanDOsampClkSupported(const char *physicalChannel[], bool32 *data);
Purpose

DAQmxGetPhysicalChanDOSampClkSupported gets the Digital Output >> Timing >> Sample Clock Supported property.
Get/Set/Reset PhysicalChan_TEDS_MfgID

int32 __CFUNC DAQmxGetPhysicalChanTEDSMfgID(const char *physicalChannel[], uInt32 *data);
Purpose

DAQmxGetPhysicalChanTEDSMfgID gets the TEDS >> ManufacturerID property.
Get/Set/Reset PhysicalChan_TEDS_ModelNum

int32 __CFUNC DAQmxGetPhysicalChanTEDSModelNum(const char *physicalChannel[], uint32 *data);
Purpose

DAQmxGetPhysicalChanTEDSModelNum gets the TEDS >> Model Number property.
Get/Set/Reset PhysicalChan_TEDS_SerialNum

int32 __CFUNC DAQmxGetPhysicalChanTEDSSerialNum(const char physicalChannel[], uInt32 *data);
Purpose

DAQmxGetPhysicalChanTEDSSerialNum gets the TEDS >> Serial Number property.
Get/Set/Reset PhysicalChan_TEDS_VersionNum

int32 __CFUNC DAQmxGetPhysicalChanTEDSVersionNum(const char physicalChannel[], uInt32 *data);
Purpose

DAQmxGetPhysicalChanTEDSVersionNum gets the TEDS >> Version Number property.
Get/Set/Reset PhysicalChan_TEDS_VersionLetter

int32 __CFUNC DAQmxGetPhysicalChanTEDSVersionLetter(const char physicalChannel[], char *data, uInt32 bufferSize);
**Purpose**

Get/Set/Reset PhysicalChan_TEDEs_BitStream

int32 __CFUNC DAQmxGetPhysicalChanTEDSBitStream(const char physicalChannel[], Uint8 *data, Uint32 arraySizeInSamples);
Purpose

DAQmxGetPhysicalChanTEDSBitStream gets the TEDS >> BitStream property.
Get/Set/Reset PhysicalChan_TEDS_TemplateIDs

int32 __CFUNC DAQmxGetPhysicalChanTEDSTemplateIDs(const char physicalChannel[], uInt32 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetPhysicalChanTEDSTemplateIDs gets the TEDS >> TemplateIDs property.
Get/Set/Reset ReadRelativeTo

int32 __CFUNC DAQmxGetReadRelativeTo(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetReadRelativeTo(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetReadRelativeTo(TaskHandle taskHandle);
Purpose

DAQmxGetReadRelativeTo gets the `RelativeTo` property.
DAQmxSetReadRelativeTo sets the `RelativeTo` property.
DAQmxResetReadRelativeTo resets the `RelativeTo` property.
Get/Set/Reset Read Offset

int32 __CFUNC DAQmxGetReadOffset(TaskHandle taskHandle, int32 *data);
int32 __CFUNC DAQmxSetReadOffset(TaskHandle taskHandle, int32 data);
int32 __CFUNC DAQmxResetReadOffset(TaskHandle taskHandle);
Purpose

DAQmxGetReadOffset gets the Offset property.
DAQmxSetReadOffset sets the Offset property.
DAQmxResetReadOffset resets the Offset property.
Get/Set/Reset Read_ChannelsToRead

int32 __CFUNC DAQmxGetReadChannelsToRead(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetReadChannelsToRead(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetReadChannelsToRead(TaskHandle taskHandle);
Purpose

DAQmxGetReadChannelsToRead gets the [Channels to Read](#) property.

DAQmxSetReadChannelsToRead sets the [Channels to Read](#) property.

DAQmxResetReadChannelsToRead resets the [Channels to Read](#) property.
Get/Set/Reset Read_ReadAllAvailSamp

int32 __CFUNC DAQmxGetReadReadAllAvailSamp(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetReadReadAllAvailSamp(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetReadReadAllAvailSamp(TaskHandle taskHandle);
**Purpose**

DAQmxGetReadReadAllAvailSamp gets the Read All Available Samples property.

DAQmxSetReadReadAllAvailSamp sets the Read All Available Samples property.

DAQmxResetReadReadAllAvailSamp resets the Read All Available Samples property.
Get/Set/Reset Read_AutoStart

int32 __CFUNC DAQmxGetReadAutoStart(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetReadAutoStart(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetReadAutoStart(TaskHandle taskHandle);
Purpose

DAQmxGetReadAutoStart gets the Auto Start property.
DAQmxSetReadAutoStart sets the Auto Start property.
DAQmxResetReadAutoStart resets the Auto Start property.
Get/Set/Reset Read_OverWrite

int32 __CFUNC DAQmxGetReadOverWrite(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetReadOverWrite(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetReadOverWrite(TaskHandle taskHandle);
Purpose

DAQmxGetReadOverWrite gets the OverWrite Mode property.
DAQmxSetReadOverWrite sets the OverWrite Mode property.
DAQmxResetReadOverWrite resets the OverWrite Mode property.
Get/Set/Reset Read_CurrReadPos

int32 __CFUNC DAQmxGetReadCurrReadPos(TaskHandle taskHandle, uInt64 *data);
**Purpose**

DAQmxGetReadCurrReadPos gets the [Status >> Current Read Position](#) property.
Get/Set/Reset Read_AvailSampPerChan

int32 __CFUNC DAQmxGetReadAvailSampPerChan(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetReadAvailSampPerChan gets the `Status >> Available Samples Per Channel` property.
Get/Set/Reset Read_TotalSampPerChanAcquired

int32 __CFUNC DAQmxGetReadTotalSampPerChanAcquired(TaskHandle taskHandle, uInt64 *data);
Purpose

DAQmxGetReadTotalSampPerChanAcquired gets the Status >> Total Samples Per Channel Acquired property.
Get/Set/Reset Read_OvercurrentChansExist

int32 __CFUNC DAQmxGetReadOvercurrentChansExist(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetReadOvercurrentChansExist gets the `Status >> Overcurrent >> Overcurrent Channels Exist` property.
Get/Set/Reset Read_OvercurrentChans

int32 __CFUNC DAQmxGetReadOvercurrentChans(TaskHandle taskHandle,
char *data, uInt32 bufferSize);
Purpose

DAQmxGetReadOvercurrentChans gets the Status >> Overcurrent >> Overcurrent Channels property.
Get/Set/Reset Read_OpenCurrentLoopChansExist

int32 __CFUNC DAQmxGetReadOpenCurrentLoopChansExist(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetReadOpenCurrentLoopChansExist gets the `Status >> Open Current Loop >> Open Current Loop Channels Exist` property.
Get/Set/Reset Read_OpenCurrentLoopChans

int32 __CFUNC DAQmxGetReadOpenCurrentLoopChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);
**Purpose**

DAQmxGetReadOpenCurrentLoopChans gets the `Status >> Open Current Loop >> Open Current Loop Channels` property.
Get/Set/Reset Read_OverloadedChansExist

int32 __CFUNC DAQmxGetReadOverloadedChansExist(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetReadOverloadedChansExist gets the Status >> Overload >> Overloaded Channels Exist property.
Get/Set/Reset Read_OverloadedChans

int32 __CFUNC DAQmxGetReadOverloadedChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);
Purpose

DAQmxGetReadOverloadedChans gets the Status >> Overload >> Overloaded Channels property.
Get/Set/Reset Read_ChangeDetect_HasOverflowed

int32 __CFUNC DAQmxGetReadChangeDetectHasOverflowed(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetReadChangeDetectHasOverflowed gets the `Status >> Advanced >> ChangeDetection >> Overflowed` property.
Get/Set/Reset Read_RawDataWidth

int32 __CFUNC DAQmxGetReadRawDataWidth(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetReadRawDataWidth gets the Advanced >> Raw Data Width property.
Get/Set/Reset Read_NumChans

int32 __CFUNC DAQmxGetReadNumChans(TaskHandle taskHandle, uInt32 *data);
**Purpose**

DAQmxGetReadNumChans gets the [Advanced >> Number of Channels](#) property.
Get/Set/Reset Read_DigitalLines_BytesPerChan

int32 __CFUNC DAQmxGetReadDigitalLinesBytesPerChan(TaskHandle taskHandle, uInt32 *data);
**Purpose**

DAQmxGetReadDigitalLinesBytesPerChan gets the [Advanced >> Digital Input >> Number of Bytes Per Channel](https://example.com) property.
Get/Set/Reset Read_WaitMode

```c
int32 __CFUNC DAQmxGetReadWaitMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetReadWaitMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetReadWaitMode(TaskHandle taskHandle);
```
Purpose

DAQmxGetReadWaitMode gets the Advanced >> Wait Mode property.
DAQmxSetReadWaitMode sets the Advanced >> Wait Mode property.
DAQmxResetReadWaitMode resets the Advanced >> Wait Mode property.
Get/Set/Reset Read_SleepTime

int32 __CFUNC DAQmxGetReadSleepTime(TaskHandle taskHandle, float64 *data);
int32 __CFUNC DAQmxSetReadSleepTime(TaskHandle taskHandle, float64 data);
int32 __CFUNC DAQmxResetReadSleepTime(TaskHandle taskHandle);
Purpose

DAQmxGetReadSleepTime gets the Advanced >> Sleep Time property.

DAQmxSetReadSleepTime sets the Advanced >> Sleep Time property.

DAQmxResetReadSleepTime resets the Advanced >> Sleep Time property.
Get/Set/Reset RealTime_ConvLateErrorsToWarnings

int32 __CFUNC DAQmxGetRealTimeConvLateErrorsToWarnings(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetRealTimeConvLateErrorsToWarnings(TaskHandle taskHandle, bool32 data);

int32 __CFUNC
    DAQmxResetRealTimeConvLateErrorsToWarnings(TaskHandle taskHandle);
Purpose

DAQmxGetRealTimeConvLateErrorsToWarnings gets the Convert Late Errors To Warnings property.

DAQmxSetRealTimeConvLateErrorsToWarnings sets the Convert Late Errors To Warnings property.

DAQmxResetRealTimeConvLateErrorsToWarnings resets the Convert Late Errors To Warnings property.
Get/Set/Reset RealTime_NumOfWarmupIters

int32 __CFUNC DAQmxGetRealTimeNumOfWarmupIters(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetRealTimeNumOfWarmupIters(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetRealTimeNumOfWarmupIters(TaskHandle taskHandle);
Purpose

DAQmxGetRealTimeNumOfWarmupIters gets the **Number Of Warmup Iterations** property.

DAQmxSetRealTimeNumOfWarmupIters sets the **Number Of Warmup Iterations** property.

DAQmxResetRealTimeNumOfWarmupIters resets the **Number Of Warmup Iterations** property.
Get/Set/Reset
RealTime_WaitForNextSampClkWaitMode

int32 __CFUNC
    DAQmxGetRealTimeWaitForNextSampClkWaitMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC
    DAQmxSetRealTimeWaitForNextSampClkWaitMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC
    DAQmxResetRealTimeWaitForNextSampClkWaitMode(TaskHandle taskHandle);
Purpose

DAQmxGetRealTimeWaitForNextSampClkWaitMode gets the **Wait For Next Sample Clock** property.

DAQmxSetRealTimeWaitForNextSampClkWaitMode sets the **Wait For Next Sample Clock** property.

DAQmxResetRealTimeWaitForNextSampClkWaitMode resets the **Wait For Next Sample Clock** property.
Get/Set/Reset RealTime_ReportMissedSamp

int32 __CFUNC DAQmxGetRealTimeReportMissedSamp(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetRealTimeReportMissedSamp(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetRealTimeReportMissedSamp(TaskHandle taskHandle);
Purpose

DAQmxGetRealTimeReportMissedSamp gets the Report Missed Samples property.
DAQmxSetRealTimeReportMissedSamp sets the Report Missed Samples property.
DAQmxResetRealTimeReportMissedSamp resets the Report Missed Samples property.
Get/Set/Reset RealTime_WriteRecoveryMode

int32 __CFUNC DAQmxGetRealTimeWriteRecoveryMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetRealTimeWriteRecoveryMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetRealTimeWriteRecoveryMode(TaskHandle taskHandle);
Purpose

DAQmxGetRealTimeWriteRecoveryMode gets the Write Recovery Mode property.

DAQmxSetRealTimeWriteRecoveryMode sets the Write Recovery Mode property.

DAQmxResetRealTimeWriteRecoveryMode resets the Write Recovery Mode property.
Get/Set/Reset Scale_Descr

int32 __CFUNC DAQmxGetScaleDescr(const char scaleName[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetScaleDescr(const char scaleName[], const char *data);
Purpose

DAQmxGetScaleDescr gets the Description property.
DAQmxSetScaleDescr sets the Description property.
Get/Set/Reset Scale_ScaledUnits

int32 __CFUNC DAQmxGetScaleScaledUnits(const char scaleName[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetScaleScaledUnits(const char scaleName[], const char *data);
Purpose

DAQmxGetScaleScaledUnits gets the Scaled Units property.

DAQmxSetScaleScaledUnits sets the Scaled Units property.
Get/Set/Reset Scale_PreScaledUnits

int32 __CFUNC DAQmxGetScalePreScaledUnits(const char scaleName[], int32 *data);

int32 __CFUNC DAQmxSetScalePreScaledUnits(const char scaleName[], int32 data);
**Purpose**

DAQmxGetScalePreScaledUnits gets the [Pre-Scaled Units](#) property.

DAQmxSetScalePreScaledUnits sets the [Pre-Scaled Units](#) property.
Get/Set/Reset Scale_Type

int32 __CFUNC DAQmxGetScaleType(const char scaleName[], int32 *data);
Purpose

DAQmxGetScaleType gets the `Scale Type` property.
Get/Set/Reset Scale_Lin_Slope

int32 __CFUNC DAQmxGetScaleLinSlope(const char scaleName[], float64 *data);

int32 __CFUNC DAQmxSetScaleLinSlope(const char scaleName[], float64 data);
Purpose

DAQmxGetScaleLinSlope gets the Linear >> Slope property.
DAQmxSetScaleLinSlope sets the Linear >> Slope property.
Get/Set/Reset Scale_Lin_YIntercept

int32 __CFUNC DAQmxGetScaleLinYIntercept(const char *scaleName[], float64 *data);

int32 __CFUNC DAQmxSetScaleLinYIntercept(const char *scaleName[], float64 data);
Purpose

DAQmxGetScaleLinYIntercept gets the Linear >> Y-Intercept property.
DAQmxSetScaleLinYIntercept sets the Linear >> Y-Intercept property.
Get/Set/Reset Scale_Map_ScaledMax

```c
int32 __CFUNC DAQmxGetScaleMapScaledMax(const char scaleName[], float64 *data);

int32 __CFUNC DAQmxSetScaleMapScaledMax(const char scaleName[], float64 data);
```
### Purpose

DAQmxGetScaleMapScaledMax gets the Map >> Scaled Maximum Value property. DAQmxSetScaleMapScaledMax sets the Map >> Scaled Maximum Value property.
Get/Set/Reset Scale_Map_PreScaledMax

int32 __CFUNC DAQmxGetScaleMapPreScaledMax(const char scaleName[], float64 *data);

int32 __CFUNC DAQmxSetScaleMapPreScaledMax(const char scaleName[], float64 data);
Purpose

DAQmxGetScaleMapPreScaledMax gets the Map >> Pre-Scaled Maximum Value property.
DAQmxSetScaleMapPreScaledMax sets the Map >> Pre-Scaled Maximum Value property.
Get/Set/Reset Scale_Map_ScaledMin

int32 __CFUNC DAQmxGetScaleMapScaledMin(const char scaleName[], float64 *data);

int32 __CFUNC DAQmxSetScaleMapScaledMin(const char scaleName[], float64 data);
Purpose

DAQmxGetScaleMapScaledMin gets the Map >> Scaled Minimum Value property.
DAQmxSetScaleMapScaledMin sets the Map >> Scaled Minimum Value property.
Get/Set/Reset Scale_Map_PreScaledMin

int32 __CFUNC DAQmxGetScaleMapPreScaledMin(const char scaleName[], float64 *data);

int32 __CFUNC DAQmxSetScaleMapPreScaledMin(const char scaleName[], float64 data);
Purpose

DAQmxGetScaleMapPreScaledMin gets the Map >> Pre-Scaled Minimum Value property.
DAQmxSetScaleMapPreScaledMin sets the Map >> Pre-Scaled Minimum Value property.
Get/Set/Reset Scale_Poly_ForwardCoeff

int32 __CFUNC DAQmxGetScalePolyForwardCoeff(const char scaleName[],
    float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetScalePolyForwardCoeff(const char scaleName[],
    float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetScalePolyForwardCoeff gets the Polynomial >> Forward Coefficients property.
DAQmxSetScalePolyForwardCoeff sets the Polynomial >> Forward Coefficients property.
Get/Set/Reset Scale Poly ReverseCoeff

int32 __CFUNC DAQmxGetScalePolyReverseCoeff(const char* scaleName[],
float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetScalePolyReverseCoeff(const char scaleName[],
float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetScalePolyReverseCoeff gets the Polynomial >> Reverse Coefficients property.
DAQmxSetScalePolyReverseCoeff sets the Polynomial >> Reverse Coefficients property.
Get/Set/Reset Scale_Table_ScaledVals

int32 __CFUNC DAQmxGetScaleTableScaledVals(const char scaleName[],
    float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetScaleTableScaledVals(const char scaleName[],
    float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetScaleTableScaledVals gets the Table >> Scaled Values property.
DAQmxSetScaleTableScaledVals sets the Table >> Scaled Values property.
Get/Set/Reset Scale_Table_PreScaledVals

int32 __CFUNC DAQmxGetScaleTablePreScaledVals(const char scaleName[],
   float64 *data, uInt32 arraySizeInSamples);

int32 __CFUNC DAQmxSetScaleTablePreScaledVals(const char scaleName[],
   float64 *data, uInt32 arraySizeInSamples);
Purpose

DAQmxGetScaleTablePreScaledVals gets the Table >> Pre-Scaled Values property.
DAQmxSetScaleTablePreScaledVals sets the Table >> Pre-Scaled Values property.
Get/Set/Reset SwitchChan_Usage

int32 __CFUNC DAQmxGetSwitchChanUsage(const char switchChannelName[], int32 *data);

int32 __CFUNC DAQmxSetSwitchChanUsage(const char switchChannelName[], int32 data);
**Purpose**

DAQmxGetSwitchChanUsage gets the **Usage** property.
DAQmxSetSwitchChanUsage sets the **Usage** property.
Get/Set/Reset SwitchChan_MaxACCarryCurrent

int32 __CFUNC DAQmxGetSwitchChanMaxACCaryCurrent(const char switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxACCarryCurrent gets the Capability >> Max AC Carry Current property.
Get/Set/Reset SwitchChan_MaxACSwitchCurrent

int32 __CFUNC DAQmxGetSwitchChanMaxACSwitchCurrent(const char switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxACSwitchCurrent gets the Capability >> Max AC Switching Current property.
Get/Set/Reset SwitchChan_MaxACCarryPwr

```c
int32 __CFUNC DAQmxGetSwitchChanMaxACCarryPwr(const char* switchChannelName[], float64 *data);
```
Purpose

DAQmxGetSwitchChanMaxACCaryPwr gets the Capability >> Max AC Carry Power property.
Get/Set/Reset SwitchChan_MaxACSwitchPwr

int32 __CFUNC DAQmxGetSwitchChanMaxACSwitchPwr(const char *switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxACSwitchPwr gets the Capability >> Max AC Switching Power property.
Get/Set/Reset SwitchChan_MaxDCCarryCurrent

int32 __CFUNC DAQmxGetSwitchChanMaxDCCarryCurrent(const char *switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxDCCarryCurrent gets the Capability >> Max DC Carry Current property.
Get/Set/Reset SwitchChan_MaxDCSwitchCurrent

int32 __CFUNC DAQmxGetSwitchChanMaxDCSwitchCurrent(const char* switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxDCSwitchCurrent gets the Capability >> Max DC Switching Current property.
Get/Set/Reset SwitchChan_MaxDCCarryPwr

int32 __CFUNC DAQmxGetSwitchChanMaxDCCarryPwr(const char* switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxDCCarryPwr gets the Capability >> Max DC Carry Power property.
Get/Set/Reset SwitchChan_MaxDCSwitchPwr

int32 __CFUNC DAQmxGetSwitchChanMaxDCSwitchPwr(const char switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxDCSwitchPwr gets the Capability >> Max DC Switching Power property.
Get/Set/Reset SwitchChan_MaxACVoltage

int32 __CFUNC DAQmxGetSwitchChanMaxACVoltage(const char switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxACVoltage gets the Capability >> Max AC Voltage property.
Get/Set/Reset SwitchChan_MaxDCVoltage

int32 __CFUNC DAQmxGetSwitchChanMaxDCVoltage(const char *switchChannelName[], float64 *data);
Purpose

DAQmxGetSwitchChanMaxDCVoltage gets the Capability >> Max DC Voltage property.
Get/Set/Reset SwitchChan_WireMode

int32 __CFUNC DAQmxGetSwitchChanWireMode(const char switchChannelName[], uInt32 *data);
Purpose

DAQmxGetSwitchChanWireMode gets the Capability >> Wire Mode property.
Get/Set/Reset SwitchChan_Bandwidth

int32 __CFUNC DAQmxGetSwitchChanBandwidth(const char *switchChannelName[], float64 *data);
**Purpose**

DAQmxGetSwitchChanBandwidth gets the [Capability >> Bandwidth](#) property.
Get/Set/Reset SwitchChan_Impedance

```c
int32 __CFUNC DAQmxGetSwitchChanImpedance(const char switchChannelName[], float64 *data);
```
Purpose

DAQmxGetSwitchChanImpedance gets the Capability >> Impedance property.
Get/Set/Reset SwitchDev_SettlingTime

int32 __CFUNC DAQmxGetSwitchDevSettlingTime(const char deviceName[], float64 *data);

int32 __CFUNC DAQmxSetSwitchDevSettlingTime(const char deviceName[], float64 data);
Purpose

DAQmxGetSwitchDevSettlingTime gets the Settling Time property.
DAQmxSetSwitchDevSettlingTime sets the Settling Time property.
Get/Set/Reset SwitchDev_AutoConnAnlgBus

```c
int32 __CFUNC DAQmxGetSwitchDevAutoConnAnlgBus(const char deviceName[], bool32 *data);

int32 __CFUNC DAQmxSetSwitchDevAutoConnAnlgBus(const char deviceName[], bool32 data);
```
Purpose

DAQmxGetSwitchDevAutoConnAnlgBus gets the [Auto Connect Analog Bus](#) property.
DAQmxSetSwitchDevAutoConnAnlgBus sets the [Auto Connect Analog Bus](#) property.
Get/Set/Reset
SwitchDev_PwrDownLatchRelaysAfterSettling

int32 __CFUNC
   DAQmxGetSwitchDevPwrDownLatchRelaysAfterSettling(const char deviceName[], bool32 *data);

int32 __CFUNC
   DAQmxSetSwitchDevPwrDownLatchRelaysAfterSettling(const char deviceName[], bool32 data);
Purpose

DAQmxGetSwitchDevPwrDownLatchRelaysAfterSettling gets the Power Down Latching property.

DAQmxSetSwitchDevPwrDownLatchRelaysAfterSettling sets the Power Down Latching property.
Get/Set/Reset SwitchDev_Settled

int32 __CFUNC DAQmxGetSwitchDevSettled(const char deviceName[], bool32 *data);
Purpose

DAQmxGetSwitchDevSettled gets the Is Settled property.
Get/Set/Reset SwitchDev_RelayList

int32 __CFUNC DAQmxGetSwitchDevRelayList(const char deviceName[], char *data, uInt32 bufferSize);
**Purpose**

DAQmxGetSwitchDevRelayList gets the Capability >> Relay List property.
Get/Set/Reset SwitchDev_NumRelays

int32 __CFUNC DAQmxGetSwitchDevNumRelays(const char deviceName[], uInt32 *data);
Purpose

DAQmxGetSwitchDevNumRelays gets the Capability >> Number of Relays property.
Get/Set/Reset SwitchDev_SwitchChanList

int32 __CFUNC DAQmxGetSwitchDevSwitchChanList(const char *deviceName[], char *data, uint32 bufferSize);
Purpose

DAQmxGetSwitchDevSwitchChanList gets the Capability >> Switch Channel List property.
Get/Set/Reset SwitchDev_NumSwitchChans

int32 __CFUNC DAQmxGetSwitchDevNumSwitchChans(const char deviceName[], uInt32 *data);
**Purpose**

DAQmxGetSwitchDevNumSwitchChans gets the Capability >> Number of Switch Channels property.
Get/Set/Reset SwitchDev_NumRows

int32 __CFUNC DAQmxGetSwitchDevNumRows(const char deviceName[], uInt32 *data);
Purpose

DAQmxGetSwitchDevNumRows gets the Capability >> Number of Rows property.
Get/Set/Reset SwitchDev_NumColumns

int32 __CFUNC DAQmxGetSwitchDevNumColumns(const char deviceName[], uInt32 *data);
Purpose

DAQmxGetSwitchDevNumColumns gets the Capability >> Number of Columns property.
Get/Set/Reset SwitchDev_Topology

int32 __CFUNC DAQmxGetSwitchDevTopology(const char deviceName[], char *data, uInt32 bufferSize);
Purpose

DAQmxGetSwitchDevTopology gets the Topology property.
Get/Set/Reset SwitchScan_BreakMode

int32 __CFUNC DAQmxGetSwitchScanBreakMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSwitchScanBreakMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSwitchScanBreakMode(TaskHandle taskHandle);
Purpose

DAQmxGetSwitchScanBreakMode gets the Break Mode property.
DAQmxSetSwitchScanBreakMode sets the Break Mode property.
DAQmxResetSwitchScanBreakMode resets the Break Mode property.
Get/Set/Reset SwitchScan_RepeaMode

int32 __CFUNC DAQmxGetSwitchScanRepeatMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSwitchScanRepeatMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSwitchScanRepeatMode(TaskHandle taskHandle);
Purpose

DAQmxGetSwitchScanRepeatMode gets the Repeat Mode property.
DAQmxSetSwitchScanRepeatMode sets the Repeat Mode property.
DAQmxResetSwitchScanRepeatMode resets the Repeat Mode property.
Get/Set/Reset SwitchScan_WaitingForAdv

int32 __CFUNC DAQmxGetSwitchScanWaitingForAdv(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetSwitchScanWaitingForAdv gets the Is Waiting For Advance property.
Get/Set/Reset Sys_GlobalChans

int32 __CFUNC DAQmxGetSysGlobalChans(char *data, uInt32 bufferSize);
**Purpose**

DAQmxGetSysGlobalChans gets the [Global Channels](Global Channels) property.
Get/Set/Reset Sys_Scales

int32 __CFUNC DAQmxGetSysScales(char *data, uInt32 bufferSize);
**Purpose**

`DAQmxGetSysScales` gets the `Scales` property.
Get/Set/Reset Sys_Tasks

int32 __CFUNC DAQmxGetSysTasks(char *data, uInt32 bufferSize);
Purpose

DAQmxGetSysTasks gets the Tasks property.
Get/Set/Reset Sys_DevNames

int32 __CFUNC DAQmxGetSysDevNames(char *data, uInt32 bufferSize);
Purpose

DAQmxGetSysDevNames gets the Device Names property.
Get/Set/Reset Sys_NIDAQMajorVersion

int32 __CFUNC DAQmxGetSysNIDAQMajorVersion(uInt32 *data);
Purpose

Get/Set/Reset Sys_NIDAQMinorVersion

int32 __CFUNC DAQmxGetSysNIDAQMinorVersion(uInt32 *data);
Purpose

Get/Set/Reset Task_Name

int32 __CFUNC DAQmxGetTaskName(TaskHandle taskHandle, char *data, uInt32 bufferSize);
Purpose

DAQmxGetTaskName gets the Name property.
Get/Set/Reset Task_Channels

int32 __CFUNC DAQmxGetTaskChannels(TaskHandle taskHandle, char *data, uInt32 bufferSize);
Purpose

DAQmxGetTaskChannels gets the Channels property.
Get/Set/Reset Task_NumChans

int32 __CFUNC DAQmxGetTaskNumChans(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetTaskNumChans gets the Number of Channels property.
Get/Set/Reset Task Devices

int32 __CFUNC DAQmxGetTaskDevices(TaskHandle taskHandle, char *data, uInt32 bufferSize);
**Purpose**

DAQmxGetTaskDevices gets the Devices property.
Get/Set/Reset Task_NumDevices

int32 __CFUNC DAQmxGetTaskNumDevices(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetTaskNumDevices gets the *Number of Devices* property.
Get/Set/Reset Task_Complete

int32 __CFUNC DAQmxGetTaskComplete(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetTaskComplete gets the Task Done property.
Get/Set/Reset SampQuant_SampMode

int32 __CFUNC DAQmxGetSampQuantSampMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSampQuantSampMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSampQuantSampMode(TaskHandle taskHandle);
Purpose

DAQmxGetSampQuantSampMode gets the Sample Quantity >> Sample Mode property.

DAQmxSetSampQuantSampMode sets the Sample Quantity >> Sample Mode property.

DAQmxResetSampQuantSampMode resets the Sample Quantity >> Sample Mode property.
Get/Set/Reset SampQuant_SampPerChan

int32 __CFUNC DAQmxGetSampQuantSampPerChan(TaskHandle taskHandle, uInt64 *data);

int32 __CFUNC DAQmxSetSampQuantSampPerChan(TaskHandle taskHandle, uInt64 data);

int32 __CFUNC DAQmxResetSampQuantSampPerChan(TaskHandle taskHandle);
Purpose

DAQmxGetSampQuantSampPerChan gets the Sample Quantity >> Samples Per Channel property.

DAQmxSetSampQuantSampPerChan sets the Sample Quantity >> Samples Per Channel property.

DAQmxResetSampQuantSampPerChan resets the Sample Quantity >> Samples Per Channel property.
Get/Set/Reset SampTimingType

```c
int32 __CFUNC DAQmxGetSampTimingType(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSampTimingType(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSampTimingType(TaskHandle taskHandle);
```
Purpose

DAQmxGetSampTimingType gets the Sample Timing Type property.

DAQmxSetSampTimingType sets the Sample Timing Type property.

DAQmxResetSampTimingType resets the Sample Timing Type property.
Get/Set/Reset SampClk_Rate

int32 __CFUNC DAQmxGetSampClkRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetSampClkRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetSampClkRate(TaskHandle taskHandle);
**Purpose**

DAQmxGetSampClkRate gets the Sample Clock >> Rate property.
DAQmxSetSampClkRate sets the Sample Clock >> Rate property.
DAQmxResetSampClkRate resets the Sample Clock >> Rate property.
Get/Set/Reset SampClk_MaxRate

int32 __CFUNC DAQmxGetSampClkMaxRate(TaskHandle taskHandle, float64 *data);
Purpose

DAQmxGetSampClkMaxRate gets the Sample Clock >> Maximum Rate property.
Get/Set/Reset SampClk_Src

int32 __CFUNC DAQmxGetSampClkSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetSampClkSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetSampClkSrc(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkSrc gets the Sample Clock >> Source property.
DAQmxSetSampClkSrc sets the Sample Clock >> Source property.
DAQmxResetSampClkSrc resets the Sample Clock >> Source property.
Get/Set/Reset SampClk_ActiveEdge

int32 __CFUNC DAQmxGetSampClkActiveEdge(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSampClkActiveEdge(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSampClkActiveEdge(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkActiveEdge gets the Sample Clock >> Active Edge property.
DAQmxSetSampClkActiveEdge sets the Sample Clock >> Active Edge property.
DAQmxResetSampClkActiveEdge resets the Sample Clock >> Active Edge property.
Get/Set/Reset SampClk_UnderflowBehavior

int32 __CFUNC DAQmxGetSampClkUnderflowBehavior(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSampClkUnderflowBehavior(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSampClkUnderflowBehavior(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkUnderflowBehavior gets the Sample Clock >> Underflow Behavior property.

DAQmxSetSampClkUnderflowBehavior sets the Sample Clock >> Underflow Behavior property.

DAQmxResetSampClkUnderflowBehavior resets the Sample Clock >> Underflow Behavior property.
Get/Set/Reset SampClk_TimebaseDiv

int32 __CFUNC DAQmxGetSampClkTimebaseDiv(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetSampClkTimebaseDiv(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetSampClkTimebaseDiv(TaskHandle taskHandle);
**Purpose**

DAQmxGetSampClkTimebaseDiv gets the [Sample Clock >> Timebase Divisor](#) property.

DAQmxSetSampClkTimebaseDiv sets the [Sample Clock >> Timebase Divisor](#) property.

DAQmxResetSampClkTimebaseDiv resets the [Sample Clock >> Timebase Divisor](#) property.
Get/Set/Reset SampClk_Timebase_Rate

int32 __CFUNC DAQmxGetSampClkTimebaseRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetSampClkTimebaseRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetSampClkTimebaseRate(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkTimebaseRate gets the Sample Clock >> Timebase >> Rate property.
DAQmxSetSampClkTimebaseRate sets the Sample Clock >> Timebase >> Rate property.
DAQmxResetSampClkTimebaseRate resets the Sample Clock >> Timebase >> Rate property.
Get/Set/Reset SampClk_Timebase_Src

int32 __CFUNC DAQmxGetSampClkTimebaseSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetSampClkTimebaseSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetSampClkTimebaseSrc(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkTimebaseSrc gets the Sample Clock >> Timebase >> Source property.
DAQmxSetSampClkTimebaseSrc sets the Sample Clock >> Timebase >> Source property.
DAQmxResetSampClkTimebaseSrc resets the Sample Clock >> Timebase >> Source property.
Get/Set/Reset SampClk_Timebase_ActiveEdge

int32 __CFUNC DAQmxGetSampClkTimebaseActiveEdge(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetSampClkTimebaseActiveEdge(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetSampClkTimebaseActiveEdge(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkTimebaseActiveEdge gets the [Sample Clock >> Timebase >> Active Edge] property.

DAQmxSetSampClkTimebaseActiveEdge sets the [Sample Clock >> Timebase >> Active Edge] property.

DAQmxResetSampClkTimebaseActiveEdge resets the [Sample Clock >> Timebase >> Active Edge] property.
Get/Set/Reset
SampClk_Timebase_MasterTimebaseDiv

int32 __CFUNC
    DAQmxGetSampClkTimebaseMasterTimebaseDiv(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetSampClkTimebaseMasterTimebaseDiv(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC
    DAQmxResetSampClkTimebaseMasterTimebaseDiv(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkTimebaseMasterTimebaseDiv gets the Sample Clock >> Timebase >> Master Timebase Divisor property.

DAQmxSetSampClkTimebaseMasterTimebaseDiv sets the Sample Clock >> Timebase >> Master Timebase Divisor property.

DAQmxResetSampClkTimebaseMasterTimebaseDiv resets the Sample Clock >> Timebase >> Master Timebase Divisor property.
Get/Set/Reset SampClk_DigFltr_Enable

int32 __CFUNC DAQmxGetSampClkDigFltrEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetSampClkDigFltrEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetSampClkDigFltrEnable(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkDigFltrEnable gets the Sample Clock >> Digital Filter >> Enable property.

DAQmxSetSampClkDigFltrEnable sets the Sample Clock >> Digital Filter >> Enable property.

DAQmxResetSampClkDigFltrEnable resets the Sample Clock >> Digital Filter >> Enable property.
Get/Set/Reset SampClk_DigFltr_MinPulseWidth

int32 __CFUNC DAQmxGetSampClkDigFltrMinPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetSampClkDigFltrMinPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetSampClkDigFltrMinPulseWidth(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkDigFltrMinPulseWidth gets the Sample Clock >> Digital Filter >> Minimum Pulse Width property.

DAQmxSetSampClkDigFltrMinPulseWidth sets the Sample Clock >> Digital Filter >> Minimum Pulse Width property.

DAQmxResetSampClkDigFltrMinPulseWidth resets the Sample Clock >> Digital Filter >> Minimum Pulse Width property.
Get/Set/Reset SampClk_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetSampClkDigFltrTimebaseSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetSampClkDigFltrTimebaseSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetSampClkDigFltrTimebaseSrc(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkDigFltrTimebaseSrc gets the Sample Clock >> Digital Filter >> Timebase >> Source property.

DAQmxSetSampClkDigFltrTimebaseSrc sets the Sample Clock >> Digital Filter >> Timebase >> Source property.

DAQmxResetSampClkDigFltrTimebaseSrc resets the Sample Clock >> Digital Filter >> Timebase >> Source property.
Get/Set/Reset SampClk_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetSampClkDigFltrTimebaseRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetSampClkDigFltrTimebaseRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetSampClkDigFltrTimebaseRate(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkDigFltrTimebaseRate gets the Sample Clock >> Digital Filter >> Timebase >> Rate property.

DAQmxSetSampClkDigFltrTimebaseRate sets the Sample Clock >> Digital Filter >> Timebase >> Rate property.

DAQmxResetSampClkDigFltrTimebaseRate resets the Sample Clock >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset SampClk_DigSync_Enable

int32 __CFUNC DAQmxGetSampClkDigSyncEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetSampClkDigSyncEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetSampClkDigSyncEnable(TaskHandle taskHandle);
Purpose

DAQmxGetSampClkDigSyncEnable gets the Sample Clock >> Digital Synchronization >> Enable property.

DAQmxSetSampClkDigSyncEnable sets the Sample Clock >> Digital Synchronization >> Enable property.

DAQmxResetSampClkDigSyncEnable resets the Sample Clock >> Digital Synchronization >> Enable property.
Get/Set/Reset Hshk_DelayAfterXfer

int32 __CFUNC DAQmxGetHshkDelayAfterXfer(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetHshkDelayAfterXfer(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetHshkDelayAfterXfer(TaskHandle taskHandle);
**Purpose**

DAQmxGetHshkDelayAfterXfer gets the Handshake >> Delay After Transfer property.

DAQmxSetHshkDelayAfterXfer sets the Handshake >> Delay After Transfer property.

DAQmxResetHshkDelayAfterXfer resets the Handshake >> Delay After Transfer property.
Get/Set/Reset Hshk_StartCond

int32 __CFUNC DAQmxGetHshkStartCond(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetHshkStartCond(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetHshkStartCond(TaskHandle taskHandle);
**Purpose**

DAQmxGetHshkStartCond gets the [Handshake >> Start Condition](#) property.

DAQmxSetHshkStartCond sets the [Handshake >> Start Condition](#) property.

DAQmxResetHshkStartCond resets the [Handshake >> Start Condition](#) property.
**Get/Set/Reset Hshk_SampleInputDataWhen**

```c
int32 __CFUNC DAQmxGetHshkSampleInputDataWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetHshkSampleInputDataWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetHshkSampleInputDataWhen(TaskHandle taskHandle);
```
Purpose

DAQmxGetHshkSampleInputDataWhen gets the Handshake >> Sample Input Data When property.

DAQmxSetHshkSampleInputDataWhen sets the Handshake >> Sample Input Data When property.

DAQmxResetHshkSampleInputDataWhen resets the Handshake >> Sample Input Data When property.
**Get/Set/Reset**  
**ChangeDetect_DI_RisingEdgePhysicalChans**  

```c
int32 __CFUNC
    DAQmxGetChangeDetectDIRisingEdgePhysicalChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetChangeDetectDIRisingEdgePhysicalChans(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetChangeDetectDIRisingEdgePhysicalChans(TaskHandle taskHandle);
```
**Purpose**

DAQmxGetChangeDetectDIRisingEdgePhysicalChans gets the Change Detection >> Digital Input >> Rising Edge Physical Channels property.

DAQmxSetChangeDetectDIRisingEdgePhysicalChans sets the Change Detection >> Digital Input >> Rising Edge Physical Channels property.

DAQmxResetChangeDetectDIRisingEdgePhysicalChans resets the Change Detection >> Digital Input >> Rising Edge Physical Channels property.
Get/Set/Reset
ChangeDetect_DI_FallingEdgePhysicalChans

int32 __CFUNC
    DAQmxGetChangeDetectDIFallingEdgePhysicalChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetChangeDetectDIFallingEdgePhysicalChans(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetChangeDetectDIFallingEdgePhysicalChans(TaskHandle taskHandle);
**Purpose**

DAQmxGetChangeDetectDIFallingEdgePhysicalChans gets the Change Detection >> Digital Input >> Falling Edge Physical Channels property.

DAQmxSetChangeDetectDIFallingEdgePhysicalChans sets the Change Detection >> Digital Input >> Falling Edge Physical Channels property.

DAQmxResetChangeDetectDIFallingEdgePhysicalChans resets the Change Detection >> Digital Input >> Falling Edge Physical Channels property.
Get/Set/Reset OnDemand_SimultaneousAOEnable

int32 __CFUNC DAQmxGetOnDemandSimultaneousAOEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetOnDemandSimultaneousAOEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetOnDemandSimultaneousAOEnable(TaskHandle taskHandle);
Purpose

DAQmxGetOnDemandSimultaneousAOEnable gets the On Demand >> Simultaneous Analog Output Enable property.

DAQmxSetOnDemandSimultaneousAOEnable sets the On Demand >> Simultaneous Analog Output Enable property.

DAQmxResetOnDemandSimultaneousAOEnable resets the On Demand >> Simultaneous Analog Output Enable property.
Get/Set/Reset AIConv_Rate

int32 __CFUNC DAQmxGetAIConvRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAIConvRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAIConvRate(TaskHandle taskHandle);
Purpose

DAQmxGetAIConvRate gets the More >> AI Convert >> Rate property.
DAQmxSetAIConvRate sets the More >> AI Convert >> Rate property.
DAQmxResetAIConvRate resets the More >> AI Convert >> Rate property.
Get/Set/Reset AICnv_Rate

int32 __CFUNC DAQmxGetAICnvRateEx(TaskHandle taskHandle, const char *deviceNames[], float64 *data);

int32 __CFUNC DAQmxSetAICnvRateEx(TaskHandle taskHandle, const char *deviceNames[], float64 data);

int32 __CFUNC DAQmxResetAICnvRateEx(TaskHandle taskHandle, const char *deviceNames[]);
Purpose

DAQmxGetAIConvRateEx gets the [More >> AI Convert >> Rate] property.
DAQmxSetAIConvRateEx sets the [More >> AI Convert >> Rate] property.
DAQmxResetAIConvRateEx resets the [More >> AI Convert >> Rate] property.
Get/Set/Reset AICnv_MaxRate

```c
int32 __CFUNC DAQmxGetAICnvMaxRate(TaskHandle taskHandle, float64 *data);
```
Purpose

DAQmxGetAIConvMaxRate gets the More >> AI Convert >> Maximum Rate property.
Get/Set/Reset AIConv_MaxRate

int32 __CFUNC DAQmxGetAIConvMaxRateEx(TaskHandle taskHandle, const char deviceNames[], float64 *data);
Purpose

DAQmxGetAICnvMaxRateEx gets the More >> AI Convert >> Maximum Rate property.
**Get/Set/Reset AIConv_Src**

```c
int32 __CFUNC DAQmxGetAIConvSrc(TaskHandle taskHandle, char *data, uint32 bufferSize);

int32 __CFUNC DAQmxSetAIConvSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAIConvSrc(TaskHandle taskHandle);
```
Purpose

DAQmxGetAIConvSrc gets the More >> Al Convert >> Source property.
DAQmxSetAIConvSrc sets the More >> Al Convert >> Source property.
DAQmxResetAIConvSrc resets the More >> Al Convert >> Source property.
Get/Set/Reset AIConv_Src

int32 __CFUNC DAQmxGetAIConvSrcEx(TaskHandle taskHandle, const char deviceNames[], char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAIConvSrcEx(TaskHandle taskHandle, const char deviceNames[], const char *data);

int32 __CFUNC DAQmxResetAIConvSrcEx(TaskHandle taskHandle, const char deviceNames[]);
Purpose

DAQmxGetAIConvSrcEx gets the More >> AI Convert >> Source property.

DAQmxSetAIConvSrcEx sets the More >> AI Convert >> Source property.

DAQmxResetAIConvSrcEx resets the More >> AI Convert >> Source property.
Get/Set/Reset AIConv_ActiveEdge

int32 __CFUNC DAQmxGetAIConvActiveEdge(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAIConvActiveEdge(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAIConvActiveEdge(TaskHandle taskHandle);
Purpose

DAQmxGetAIConvActiveEdge gets the More >> AI Convert >> Active Edge property.
DAQmxSetAIConvActiveEdge sets the More >> AI Convert >> Active Edge property.
DAQmxResetAIConvActiveEdge resets the More >> AI Convert >> Active Edge property.
Get/Set/Reset AIConv_ActiveEdge

```c
int32 __CFUNC DAQmxGetAIConvActiveEdgeEx(TaskHandle taskHandle,
                           const char deviceNames[], int32 *data);

int32 __CFUNC DAQmxSetAIConvActiveEdgeEx(TaskHandle taskHandle,
                           const char deviceNames[], int32 data);

int32 __CFUNC DAQmxResetAIConvActiveEdgeEx(TaskHandle taskHandle,
                           const char deviceNames[]);
```
Purpose

DAQmxGetAIConvActiveEdgeEx gets the More >> AI Convert >> Active Edge property.

DAQmxSetAIConvActiveEdgeEx sets the More >> AI Convert >> Active Edge property.

DAQmxResetAIConvActiveEdgeEx resets the More >> AI Convert >> Active Edge property.
Get/Set/Reset AICnv_TimebaseDiv

int32 __CFUNC DAQmxGetAICnvTimebaseDiv(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetAICnvTimebaseDiv(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetAICnvTimebaseDiv(TaskHandle taskHandle);
Purpose

DAQmxGetAIConvTimebaseDiv gets the More >> AI Convert >> Timebase Divisor property.
DAQmxSetAIConvTimebaseDiv sets the More >> AI Convert >> Timebase Divisor property.
DAQmxResetAIConvTimebaseDiv resets the More >> AI Convert >> Timebase Divisor property.
Get/Set/Reset AI\texttt{Conv}\_TimebaseDiv

\begin{verbatim}
int32 __CFUNC DAQmxGetAIConvTimebaseDivEx(TaskHandle taskHandle, 
const char deviceNames[], uInt32 *data);

int32 __CFUNC DAQmxSetAIConvTimebaseDivEx(TaskHandle taskHandle, 
const char deviceNames[], uInt32 data);

int32 __CFUNC DAQmxResetAIConvTimebaseDivEx(TaskHandle taskHandle, 
const char deviceNames[]);
\end{verbatim}
Purpose

DAQmxGetAIConvTimebaseDivEx gets the More >> AI Convert >> Timebase Divisor property.
DAQmxSetAIConvTimebaseDivEx sets the More >> AI Convert >> Timebase Divisor property.
DAQmxResetAIConvTimebaseDivEx resets the More >> AI Convert >> Timebase Divisor property.
Get/Set/Reset AIConv_Timebase_Ptr

int32 __CFUNC DAQmxGetAIConvTimebaseSrc(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAIConvTimebaseSrc(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAIConvTimebaseSrc(TaskHandle taskHandle);
**Purpose**

DAQmxGetAIConvTimebaseSrc gets the [More >> AI Convert >> Timebase >> Source](#) property.

DAQmxSetAIConvTimebaseSrc sets the [More >> AI Convert >> Timebase >> Source](#) property.

DAQmxResetAIConvTimebaseSrc resets the [More >> AI Convert >> Timebase >> Source](#) property.
Get/Set/Reset AICnv_Timebase_Src

int32 __CFUNC DAQmxGetAICnvTimebaseSrcEx(TaskHandle taskHandle,
const char deviceNames[], int32 *data);

int32 __CFUNC DAQmxSetAICnvTimebaseSrcEx(TaskHandle taskHandle,
const char deviceNames[], int32 data);

int32 __CFUNC DAQmxResetAICnvTimebaseSrcEx(TaskHandle taskHandle,
const char deviceNames[]);
Purpose


DAQmxSetAIConvTimebaseSrcEx sets the [More >> AI Convert >> Timebase >> Source] property.

DAQmxResetAIConvTimebaseSrcEx resets the [More >> AI Convert >> Timebase >> Source] property.
Get/Set/Reset DelayFromSampClk_DelayUnits

int32 __CFUNC DAQmxGetDelayFromSampClkDelayUnits(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDelayFromSampClkDelayUnits(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDelayFromSampClkDelayUnits(TaskHandle taskHandle);
Purpose

DAQmxGetDelayFromSampClkDelayUnits gets the More >> AI Convert >> Delay From Sample Clock >> Delay Units property.

DAQmxSetDelayFromSampClkDelayUnits sets the More >> AI Convert >> Delay From Sample Clock >> Delay Units property.

DAQmxResetDelayFromSampClkDelayUnits resets the More >> AI Convert >> Delay From Sample Clock >> Delay Units property.
Get/Set/Reset DelayFromSampClk_DelayUnits

int32 __CFUNC DAQmxGetDelayFromSampClkDelayUnitsEx(TaskHandle taskHandle, const char deviceNames[], int32 *data);

int32 __CFUNC DAQmxSetDelayFromSampClkDelayUnitsEx(TaskHandle taskHandle, const char deviceNames[], int32 data);

int32 __CFUNC DAQmxResetDelayFromSampClkDelayUnitsEx(TaskHandle taskHandle, const char deviceNames[]);
Purpose

DAQmxGetDelayFromSampClkDelayUnitsEx gets the More >> AI Convert >> Delay From Sample Clock >> Delay Units property.

DAQmxSetDelayFromSampClkDelayUnitsEx sets the More >> AI Convert >> Delay From Sample Clock >> Delay Units property.

DAQmxResetDelayFromSampClkDelayUnitsEx resets the More >> AI Convert >> Delay From Sample Clock >> Delay Units property.
Get/Set/Reset DelayFromSampClk_Delay

int32 __CFUNC DAQmxGetDelayFromSampClkDelay(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetDelayFromSampClkDelay(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetDelayFromSampClkDelay(TaskHandle taskHandle);
Purpose

DAQmxGetDelayFromSampClkDelay gets the Delay property.

DAQmxSetDelayFromSampClkDelay sets the Delay property.

DAQmxResetDelayFromSampClkDelay resets the Delay property.
Get/Set/Reset DelayFromSampClk_Delay

```c
int32 __CFUNC DAQmxGetDelayFromSampClkDelayEx(TaskHandle taskHandle, const char deviceNames[], float64 *data);

int32 __CFUNC DAQmxSetDelayFromSampClkDelayEx(TaskHandle taskHandle, const char deviceNames[], float64 data);

int32 __CFUNC DAQmxResetDelayFromSampClkDelayEx(TaskHandle taskHandle, const char deviceNames[]);
```
Purpose

DAQmxGetDelayFromSampClkDelayEx gets the Delay property.

DAQmxSetDelayFromSampClkDelayEx sets the Delay property.

DAQmxResetDelayFromSampClkDelayEx resets the Delay property.
Get/Set/Reset MasterTimebase_Rate

int32 __CFUNC DAQmxGetMasterTimebaseRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetMasterTimebaseRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetMasterTimebaseRate(TaskHandle taskHandle);
Purpose

DAQmxGetMasterTimebaseRate gets the More >> Master Timebase >> Rate property.
DAQmxSetMasterTimebaseRate sets the More >> Master Timebase >> Rate property.
DAQmxResetMasterTimebaseRate resets the More >> Master Timebase >> Rate property.
Get/Set/Reset MasterTimebase_Src

int32 __CFUNC DAQmxGetMasterTimebaseSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetMasterTimebaseSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetMasterTimebaseSrc(TaskHandle taskHandle);
Purpose

DAQmxGetMasterTimebaseSrc gets the More >> Master Timebase >> Source property.

DAQmxSetMasterTimebaseSrc sets the More >> Master Timebase >> Source property.

DAQmxResetMasterTimebaseSrc resets the More >> Master Timebase >> Source property.
Get/Set/Reset RefClk_Rate

int32 __CFUNC DAQmxGetRefClkRate(TaskHandle taskHandle, float64 *data);
int32 __CFUNC DAQmxSetRefClkRate(TaskHandle taskHandle, float64 data);
int32 __CFUNC DAQmxResetRefClkRate(TaskHandle taskHandle);
Purpose

DAQmxGetRefClkRate gets the Reference Clock Rate property.
DAQmxSetRefClkRate sets the Reference Clock Rate property.
DAQmxResetRefClkRate resets the Reference Clock Rate property.
Get/Set/Reset RefClk_Src

int32 __CFUNC DAQmxGetRefClkSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetRefClkSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetRefClkSrc(TaskHandle taskHandle);
**Purpose**

DAQmxGetRefClkSrc gets the [More >> Reference Clock >> Source](#) property.

DAQmxSetRefClkSrc sets the [More >> Reference Clock >> Source](#) property.

DAQmxResetRefClkSrc resets the [More >> Reference Clock >> Source](#) property.
**Get/Set/Reset SyncPulse_Src**

```c
int32 __CFUNC DAQmxGetSyncPulseSrc(TaskHandle taskHandle, char *data, 
        UInt32 bufferSize);
```

```c
int32 __CFUNC DAQmxSetSyncPulseSrc(TaskHandle taskHandle, const char
        *data);
```

```c
int32 __CFUNC DAQmxResetSyncPulseSrc(TaskHandle taskHandle);
```
Purpose

DAQmxGetSyncPulseSrc gets the Synchronization Pulse Source property. 

DAQmxSetSyncPulseSrc sets the Synchronization Pulse Source property. 

DAQmxResetSyncPulseSrc resets the Synchronization Pulse Source property.
Get/Set/Reset SyncPulse_SyncTime

int32 __CFUNC DAQmxGetSyncPulseSyncTime(TaskHandle taskHandle, float64 *data);
Purpose

DAQmxGetSyncPulseSyncTime gets the property.
Get/Set/Reset SyncPulse_MinDelayToStart

int32 __CFUNC DAQmxGetSyncPulseMinDelayToStart(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetSyncPulseMinDelayToStart(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetSyncPulseMinDelayToStart(TaskHandle taskHandle);
Purpose

DAQmxGetSyncPulseMinDelayToStart gets the More >> Synchronization Pulse >> Minimum Delay To Start property.

DAQmxSetSyncPulseMinDelayToStart sets the More >> Synchronization Pulse >> Minimum Delay To Start property.

DAQmxResetSyncPulseMinDelayToStart resets the More >> Synchronization Pulse >> Minimum Delay To Start property.
Get/Set/Reset SampTimingEngine

int32 __CFUNC DAQmxGetSampTimingEngine(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetSampTimingEngine(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetSampTimingEngine(TaskHandle taskHandle);
Purpose

DAQmxGetSampTimingEngine gets the Advanced >> Sample Timing Engine property.
DAQmxSetSampTimingEngine sets the Advanced >> Sample Timing Engine property.
DAQmxResetSampTimingEngine resets the Advanced >> Sample Timing Engine property.
Get/Set/Reset StartTrig_Type

```c
int32 __CFUNC DAQmxGetStartTrigType(TaskHandle taskHandle, int32 *data);
int32 __CFUNC DAQmxSetStartTrigType(TaskHandle taskHandle, int32 data);
int32 __CFUNC DAQmxResetStartTrigType(TaskHandle taskHandle);
```
Purpose

DAQmxGetStartTrigType gets the `Start >> Trigger Type` property.
DAQmxSetStartTrigType sets the `Start >> Trigger Type` property.
DAQmxResetStartTrigType resets the `Start >> Trigger Type` property.
Get/Set/Reset DigEdge_StartTrig_Src

int32 __CFUNC DAQmxGetDigEdgeStartTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigEdgeStartTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigEdgeStartTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeStartTrigSrc gets the `Start >> Digital Edge >> Source` property.
DAQmxSetDigEdgeStartTrigSrc sets the `Start >> Digital Edge >> Source` property.
DAQmxResetDigEdgeStartTrigSrc resets the `Start >> Digital Edge >> Source` property.
Get/Set/Reset DigEdge_StartTrig_Edge

int32 __CFUNC DAQmxGetDigEdgeStartTrigEdge(TaskHandle taskHandle,
                                           int32 *data);

int32 __CFUNC DAQmxSetDigEdgeStartTrigEdge(TaskHandle taskHandle,
                                           int32 data);

int32 __CFUNC DAQmxResetDigEdgeStartTrigEdge(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeStartTrigEdge gets the Start Edge property.
DAQmxSetDigEdgeStartTrigEdge sets the Start Edge property.
DAQmxResetDigEdgeStartTrigEdge resets the Start Edge property.
Get/Set/Reset DigEdge_StartTrig_DigFltr_Enable

int32 __CFUNC DAQmxGetDigEdgeStartTrigDigFltrEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigEdgeStartTrigDigFltrEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetDigEdgeStartTrigDigFltrEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeStartTrigDigFltrEnable gets the property.

DAQmxSetDigEdgeStartTrigDigFltrEnable sets the property.

DAQmxResetDigEdgeStartTrigDigFltrEnable resets the property.
Get/Set/Reset
DigEdge_StartTrig_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetDigEdgeStartTrigDigFltrMinPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC
    DAQmxSetDigEdgeStartTrigDigFltrMinPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC
    DAQmxResetDigEdgeStartTrigDigFltrMinPulseWidth(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeStartTrigDigFltrMinPulseWidth gets the **Start Edge Digital Filter Minimum Pulse Width** property.

DAQmxSetDigEdgeStartTrigDigFltrMinPulseWidth sets the **Start Edge Digital Filter Minimum Pulse Width** property.

DAQmxResetDigEdgeStartTrigDigFltrMinPulseWidth resets the **Start Edge Digital Filter Minimum Pulse Width** property.
Get/Set/Reset
DigEdge_StartTrig_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetDigEdgeStartTrigDigFltrTimebaseSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigEdgeStartTrigDigFltrTimebaseSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetDigEdgeStartTrigDigFltrTimebaseSrc(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeStartTrigDigFltrTimebaseSrc gets the property.

DAQmxSetDigEdgeStartTrigDigFltrTimebaseSrc sets the property.

DAQmxResetDigEdgeStartTrigDigFltrTimebaseSrc resets the property.
Get/Set/Reset

**DigEdge_StartTrig_DigFltr_TimebaseRate**

```c
int32 __CFUNC
    DAQmxGetDigEdgeStartTrigDigFltrTimebaseRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetDigEdgeStartTrigDigFltrTimebaseRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC
    DAQmxResetDigEdgeStartTrigDigFltrTimebaseRate(TaskHandle taskHandle);
```
Purpose

DAQmxGetDigEdgeStartTrigDigFltrTimebaseRate gets the `Start >> Digital Edge >> Digital Filter >> Timebase >> Rate` property.

DAQmxSetDigEdgeStartTrigDigFltrTimebaseRate sets the `Start >> Digital Edge >> Digital Filter >> Timebase >> Rate` property.

DAQmxResetDigEdgeStartTrigDigFltrTimebaseRate resets the `Start >> Digital Edge >> Digital Filter >> Timebase >> Rate` property.
Get/Set/Reset DigEdge_StartTrig_DigSync_Enable

int32 __CFUNC DAQmxGetDigEdgeStartTrigDigSyncEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigEdgeStartTrigDigSyncEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetDigEdgeStartTrigDigSyncEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeStartTrigDigSyncEnable gets the Start Edge Digital Synchronization Enable property.

DAQmxSetDigEdgeStartTrigDigSyncEnable sets the Start Edge Digital Synchronization Enable property.

DAQmxResetDigEdgeStartTrigDigSyncEnable resets the Start Edge Digital Synchronization Enable property.
Get/Set/Reset DigPattern_StartTrig_Src

int32 __CFUNC DAQmxGetDigPatternStartTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigPatternStartTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigPatternStartTrigSrc(TaskHandle taskHandle);
**Purpose**

DAQmxGetDigPatternStartTrigSrc gets the [Start >> Digital Pattern >> Source](#) property.

DAQmxSetDigPatternStartTrigSrc sets the [Start >> Digital Pattern >> Source](#) property.

DAQmxResetDigPatternStartTrigSrc resets the [Start >> Digital Pattern >> Source](#) property.
Get/Set/Reset DigPattern_StartTrig_Pattern

int32 __CFUNC DAQmxGetDigPatternStartTrigPattern(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigPatternStartTrigPattern(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigPatternStartTrigPattern(TaskHandle taskHandle);
Purpose

DAQmxGetDigPatternStartTrigPattern gets the `<Start >> Digital Pattern >> Pattern>` property.
DAQmxSetDigPatternStartTrigPattern sets the `<Start >> Digital Pattern >> Pattern>` property.
DAQmxResetDigPatternStartTrigPattern resets the `<Start >> Digital Pattern >> Pattern>` property.
Get/Set/Reset DigPattern_StartTrig_When

int32 __CFUNC DAQmxGetDigPatternStartTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDigPatternStartTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDigPatternStartTrigWhen(TaskHandle taskHandle);
Purpose

DAQmxGetDigPatternStartTrigWhen gets the Start >> Digital Pattern >> Trigger When property.
DAQmxSetDigPatternStartTrigWhen sets the Start >> Digital Pattern >> Trigger When property.
DAQmxResetDigPatternStartTrigWhen resets the Start >> Digital Pattern >> Trigger When property.
Get/Set/Reset AnlgEdge_StartTrig_Src

int32 __CFUNC DAQmxGetAnlgEdgeStartTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAnlgEdgeStartTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAnlgEdgeStartTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgEdgeStartTrigSrc gets the Start >> Analog Edge >> Source property.
DAQmxSetAnlgEdgeStartTrigSrc sets the Start >> Analog Edge >> Source property.
DAQmxResetAnlgEdgeStartTrigSrc resets the Start >> Analog Edge >> Source property.
Get/Set/Reset AnlgEdge_StartTrig_Slope

int32 __CFUNC DAQmxGetAnlgEdgeStartTrigSlope(TaskHandle taskHandle,
int32 *data);

int32 __CFUNC DAQmxSetAnlgEdgeStartTrigSlope(TaskHandle taskHandle,
int32 data);

int32 __CFUNC DAQmxResetAnlgEdgeStartTrigSlope(TaskHandle
taskHandle);
Purpose

DAQmxGetAnlgEdgeStartTrigSlope gets the Start >> Analog Edge >> Slope property.

DAQmxSetAnlgEdgeStartTrigSlope sets the Start >> Analog Edge >> Slope property.

DAQmxResetAnlgEdgeStartTrigSlope resets the Start >> Analog Edge >> Slope property.
Get/Set/Reset AnlgEdge_StartTrig_Lvl

```c
int32 __CFUNC DAQmxGetAnlgEdgeStartTrigLvl(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgEdgeStartTrigLvl(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgEdgeStartTrigLvl(TaskHandle taskHandle);
```
Purpose

DAQmxGetAnlgEdgeStartTrigLvl gets the Start >> Analog Edge >> Level property.
DAQmxSetAnlgEdgeStartTrigLvl sets the Start >> Analog Edge >> Level property.
DAQmxResetAnlgEdgeStartTrigLvl resets the Start >> Analog Edge >> Level property.
Get/Set/Reset AnlgEdge_StartTrig_Hyst

int32 __CFUNC DAQmxGetAnlgEdgeStartTrigHyst(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgEdgeStartTrigHyst(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgEdgeStartTrigHyst(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgEdgeStartTrigHyst gets the Start >> Analog Edge >> Hysteresis property.
DAQmxSetAnlgEdgeStartTrigHyst sets the Start >> Analog Edge >> Hysteresis property.
DAQmxResetAnlgEdgeStartTrigHyst resets the Start >> Analog Edge >> Hysteresis property.
Get/Set/Reset AnlgEdge_StartTrig_Coupling

int32 __CFUNC DAQmxGetAnlgEdgeStartTrigCoupling(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgEdgeStartTrigCoupling(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgEdgeStartTrigCoupling(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgEdgeStartTrigCoupling gets the Start >> Analog Edge >> Coupling property.
DAQmxSetAnlgEdgeStartTrigCoupling sets the Start >> Analog Edge >> Coupling property.
DAQmxResetAnlgEdgeStartTrigCoupling resets the Start >> Analog Edge >> Coupling property.
Get/Set/Reset AnlgWin_StartTrig_Src

int32 __CFUNC DAQmxGetAnlgWinStartTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAnlgWinStartTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAnlgWinStartTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinStartTrigSrc gets the Start >> Analog Window >> Source property.
DAQmxSetAnlgWinStartTrigSrc sets the Start >> Analog Window >> Source property.
DAQmxResetAnlgWinStartTrigSrc resets the Start >> Analog Window >> Source property.
Get/Set/Reset AnlgWin_StartTrig_When

int32 __CFUNC DAQmxGetAnlgWinStartTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgWinStartTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgWinStartTrigWhen(TaskHandle taskHandle);
**Purpose**

DAQmxGetAnlgWinStartTrigWhen gets the Start >> Analog Window >> Trigger When property. DAQmxSetAnlgWinStartTrigWhen sets the Start >> Analog Window >> Trigger When property. DAQmxResetAnlgWinStartTrigWhen resets the Start >> Analog Window >> Trigger When property.
Get/Set/Reset AnlgWin_StartTrig_Top

int32 __CFUNC DAQmxGetAnlgWinStartTrigTop(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgWinStartTrigTop(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgWinStartTrigTop(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinStartTrigTop gets the Start >> Analog Window >> Top property.
DAQmxSetAnlgWinStartTrigTop sets the Start >> Analog Window >> Top property.
DAQmxResetAnlgWinStartTrigTop resets the Start >> Analog Window >> Top property.
Get/Set/Reset AnlgWin_StartTrig_Btm

int32 __CFUNC DAQmxGetAnlgWinStartTrigBtm(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgWinStartTrigBtm(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgWinStartTrigBtm(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinStartTrigBtm gets the Start >> Analog Window >> Bottom property.

DAQmxSetAnlgWinStartTrigBtm sets the Start >> Analog Window >> Bottom property.

DAQmxResetAnlgWinStartTrigBtm resets the Start >> Analog Window >> Bottom property.
Get/Set/Reset AnlgWin_StartTrig_Coupling

int32 __CFUNC DAQmxGetAnlgWinStartTrigCoupling(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgWinStartTrigCoupling(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgWinStartTrigCoupling(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinStartTrigCoupling gets the Start >> Analog Window >> Coupling property.
DAQmxSetAnlgWinStartTrigCoupling sets the Start >> Analog Window >> Coupling property.
DAQmxResetAnlgWinStartTrigCoupling resets the Start >> Analog Window >> Coupling property.
Get/Set/Reset StartTrig_Delay

int32 __CFUNC DAQmxGetStartTrigDelay(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetStartTrigDelay(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetStartTrigDelay(TaskHandle taskHandle);
Purpose

DAQmxGetStartTrigDelay gets the Start >> More >> Delay property.

DAQmxSetStartTrigDelay sets the Start >> More >> Delay property.

DAQmxResetStartTrigDelay resets the Start >> More >> Delay property.
Get/Set/Reset StartTrig_DelayUnits

int32 __CFUNC DAQmxGetStartTrigDelayUnits(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetStartTrigDelayUnits(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetStartTrigDelayUnits(TaskHandle taskHandle);
Purpose

DAQmxGetStartTrigDelayUnits gets the Start >> More >> Delay Units property.
DAQmxSetStartTrigDelayUnits sets the Start >> More >> Delay Units property.
DAQmxResetStartTrigDelayUnits resets the Start >> More >> Delay Units property.
Get/Set/Reset StartTrig_Retriggerable

int32 __CFUNC DAQmxGetStartTrigRetriggerable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetStartTrigRetriggerable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetStartTrigRetriggerable(TaskHandle taskHandle);
Purpose

DAQmxGetStartTrigRetriggerable gets the Start >> More >> Retriggerable property.
DAQmxSetStartTrigRetriggerable sets the Start >> More >> Retriggerable property.
DAQmxResetStartTrigRetriggerable resets the Start >> More >> Retriggerable property.
Get/Set/Reset RefTrig_Type

int32 __CFUNC DAQmxGetRefTrigType(TaskHandle taskHandle, int32 *data);
int32 __CFUNC DAQmxSetRefTrigType(TaskHandle taskHandle, int32 data);
int32 __CFUNC DAQmxResetRefTrigType(TaskHandle taskHandle);
Purpose

DAQmxGetRefTrigType gets the Reference>>Trigger Type property.
DAQmxSetRefTrigType sets the Reference>>Trigger Type property.
DAQmxResetRefTrigType resets the Reference>>Trigger Type property.
Get/Set/Reset RefTrig_PretrigSamples

int32 __CFUNC DAQmxGetRefTrigPretrigSamples(TaskHandle taskHandle, uInt32 *data);

int32 __CFUNC DAQmxSetRefTrigPretrigSamples(TaskHandle taskHandle, uInt32 data);

int32 __CFUNC DAQmxResetRefTrigPretrigSamples(TaskHandle taskHandle);
Purpose

DAQmxGetRefTrigPretrigSamples gets the Reference >> Pretrigger Samples per Channel property.

DAQmxSetRefTrigPretrigSamples sets the Reference >> Pretrigger Samples per Channel property.

DAQmxResetRefTrigPretrigSamples resets the Reference >> Pretrigger Samples per Channel property.
Get/Set/Reset DigEdge_RefTrig_Src

int32 __CFUNC DAQmxGetDigEdgeRefTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigEdgeRefTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigEdgeRefTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeRefTrigSrc gets the Reference >> Digital Edge >> Source property.

DAQmxSetDigEdgeRefTrigSrc sets the Reference >> Digital Edge >> Source property.

DAQmxResetDigEdgeRefTrigSrc resets the Reference >> Digital Edge >> Source property.
Get/Set/Reset DigEdge_RefTrig_Edge

int32 __CFUNC DAQmxGetDigEdgeRefTrigEdge(TaskHandle taskHandle,
int32 *data);

int32 __CFUNC DAQmxSetDigEdgeRefTrigEdge(TaskHandle taskHandle,
int32 data);

int32 __CFUNC DAQmxResetDigEdgeRefTrigEdge(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeRefTrigEdge gets the Reference >> Digital Edge >> Edge property.
DAQmxSetDigEdgeRefTrigEdge sets the Reference >> Digital Edge >> Edge property.
DAQmxResetDigEdgeRefTrigEdge resets the Reference >> Digital Edge >> Edge property.
**Get/Set/Reset DigPattern_RefTrig_Src**

```c
int32 __CFUNC DAQmxGetDigPatternRefTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigPatternRefTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigPatternRefTrigSrc(TaskHandle taskHandle);
```
**Purpose**

DAQmxGetDigPatternRefTrigSrc gets the Reference >> Digital Pattern >> Source property.

DAQmxSetDigPatternRefTrigSrc sets the Reference >> Digital Pattern >> Source property.

DAQmxResetDigPatternRefTrigSrc resets the Reference >> Digital Pattern >> Source property.
Get/Set/Reset DigPattern_RefTrig_Pattern

int32 __CFUNC DAQmxGetDigPatternRefTrigPattern(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigPatternRefTrigPattern(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigPatternRefTrigPattern(TaskHandle taskHandle);
Purpose

DAQmxGetDigPatternRefTrigPattern gets the Reference >> Digital Pattern >> Pattern property.
DAQmxSetDigPatternRefTrigPattern sets the Reference >> Digital Pattern >> Pattern property.
DAQmxResetDigPatternRefTrigPattern resets the Reference >> Digital Pattern >> Pattern property.
Get/Set/Reset DigPattern_RefTrig_When

int32 __CFUNC DAQmxGetDigPatternRefTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDigPatternRefTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDigPatternRefTrigWhen(TaskHandle taskHandle);
Purpose

DAQmxGetDigPatternRefTrigWhen gets the Reference >> Digital Pattern >> Trigger When property.

DAQmxSetDigPatternRefTrigWhen sets the Reference >> Digital Pattern >> Trigger When property.

DAQmxResetDigPatternRefTrigWhen resets the Reference >> Digital Pattern >> Trigger When property.
Get/Set/Reset AnlgEdge_RefTrig_Src

int32 __CFUNC DAQmxGetAnlgEdgeRefTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAnlgEdgeRefTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAnlgEdgeRefTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgEdgeRefTrigSrc gets the Reference >> Analog Edge >> Source property.
DAQmxSetAnlgEdgeRefTrigSrc sets the Reference >> Analog Edge >> Source property.
DAQmxResetAnlgEdgeRefTrigSrc resets the Reference >> Analog Edge >> Source property.
Get/Set/Reset AnlgEdge_RefTrig_Slope

int32 __CFUNC DAQmxGetAnlgEdgeRefTrigSlope(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgEdgeRefTrigSlope(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgEdgeRefTrigSlope(TaskHandle taskHandle);
**Purpose**

DAQmxGetAnlgEdgeRefTrigSlope gets the Reference >> Analog Edge >> Slope property.

DAQmxSetAnlgEdgeRefTrigSlope sets the Reference >> Analog Edge >> Slope property.

DAQmxResetAnlgEdgeRefTrigSlope resets the Reference >> Analog Edge >> Slope property.
Get/Set/Reset AnlgEdge_RefTrig_Lvl

int32 __CFUNC DAQmxGetAnlgEdgeRefTrigLvl(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgEdgeRefTrigLvl(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgEdgeRefTrigLvl(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgEdgeRefTrigLvl gets the Reference >> Analog Edge >> Level property.

DAQmxSetAnlgEdgeRefTrigLvl sets the Reference >> Analog Edge >> Level property.

DAQmxResetAnlgEdgeRefTrigLvl resets the Reference >> Analog Edge >> Level property.
Get/Set/Reset \texttt{AnlgEdge\_RefTrig\_Hyst}

\texttt{int32 \_\_CFUNC DAQmxGetAnlgEdgeRefTrigHyst(TaskHandle taskHandle, float64 *data);}\

\texttt{int32 \_\_CFUNC DAQmxSetAnlgEdgeRefTrigHyst(TaskHandle taskHandle, float64 data);}\

\texttt{int32 \_\_CFUNC DAQmxResetAnlgEdgeRefTrigHyst(TaskHandle taskHandle);}
**Purpose**

DAQmxGetAnlgEdgeRefTrigHyst gets the Reference >> Analog Edge >> Hysteresis property.

DAQmxSetAnlgEdgeRefTrigHyst sets the Reference >> Analog Edge >> Hysteresis property.

DAQmxResetAnlgEdgeRefTrigHyst resets the Reference >> Analog Edge >> Hysteresis property.
Get/Set/Reset AnlgEdge_RefTrig_Coupling

int32 __CFUNC DAQmxGetAnlgEdgeRefTrigCoupling(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgEdgeRefTrigCoupling(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgEdgeRefTrigCoupling(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgEdgeRefTrigCoupling gets the Reference >> Analog Edge >> Coupling property.

DAQmxSetAnlgEdgeRefTrigCoupling sets the Reference >> Analog Edge >> Coupling property.

DAQmxResetAnlgEdgeRefTrigCoupling resets the Reference >> Analog Edge >> Coupling property.
Get/Set/Reset AnlgWin_RefTrig_Src

int32 __CFUNC DAQmxGetAnlgWinRefTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAnlgWinRefTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAnlgWinRefTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinRefTrigSrc gets the Reference >> Analog Window >> Source property.
DAQmxSetAnlgWinRefTrigSrc sets the Reference >> Analog Window >> Source property.
DAQmxResetAnlgWinRefTrigSrc resets the Reference >> Analog Window >> Source property.
Get/Set/Reset AnlgWin_RefTrig_When

int32 __CFUNC DAQmxGetAnlgWinRefTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgWinRefTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgWinRefTrigWhen(TaskHandle taskHandle);
**Purpose**

DAQmxGetAnlgWinRefTrigWhen gets the [Reference >> Analog Window >> Trigger When](#) property.

DAQmxSetAnlgWinRefTrigWhen sets the [Reference >> Analog Window >> Trigger When](#) property.

DAQmxResetAnlgWinRefTrigWhen resets the [Reference >> Analog Window >> Trigger When](#) property.
Get/Set/Reset AnlgWin_RefTrig_Top

int32 __CFUNC DAQmxGetAnlgWinRefTrigTop(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgWinRefTrigTop(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgWinRefTrigTop(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinRefTrigTop gets the Reference >> Analog Window >> Top property.
DAQmxSetAnlgWinRefTrigTop sets the Reference >> Analog Window >> Top property.
DAQmxResetAnlgWinRefTrigTop resets the Reference >> Analog Window >> Top property.
Get/Set/Reset AnlgWin_RefTrig_Btm

int32 __CFUNC DAQmxGetAnlgWinRefTrigBtm(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgWinRefTrigBtm(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgWinRefTrigBtm(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinRefTrigBtm gets the Reference >> Analog Window >> Bottom property.
DAQmxSetAnlgWinRefTrigBtm sets the Reference >> Analog Window >> Bottom property.
DAQmxResetAnlgWinRefTrigBtm resets the Reference >> Analog Window >> Bottom property.
Get/Set/Reset AnlgWin_RefTrig_Coupling

int32 __CFUNC DAQmxGetAnlgWinRefTrigCoupling(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgWinRefTrigCoupling(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgWinRefTrigCoupling(TaskHandle taskHandle);
**Purpose**

DAQmxGetAnlgWinRefTrigCoupling gets the Reference >> Analog Window >> Coupling property.

DAQmxSetAnlgWinRefTrigCoupling sets the Reference >> Analog Window >> Coupling property.

DAQmxResetAnlgWinRefTrigCoupling resets the Reference >> Analog Window >> Coupling property.
Get/Set/Reset AdvTrig_Type

int32 __CFUNC DAQmxGetAdvTrigType(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAdvTrigType(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAdvTrigType(TaskHandle taskHandle);
**Purpose**

DAQmxGetAdvTrigType gets the [More >> Advance >> Trigger Type](#) property.

DAQmxSetAdvTrigType sets the [More >> Advance >> Trigger Type](#) property.

DAQmxResetAdvTrigType resets the [More >> Advance >> Trigger Type](#) property.
Get/Set/Reset DigEdge_AdvTrig_Src

int32 __CFUNC DAQmxGetDigEdgeAdvTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigEdgeAdvTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigEdgeAdvTrigSrc(TaskHandle taskHandle);
**Purpose**

DAQmxGetDigEdgeAdvTrigSrc gets the More >> Advance >> Digital Edge >> Source property.

DAQmxSetDigEdgeAdvTrigSrc sets the More >> Advance >> Digital Edge >> Source property.

DAQmxResetDigEdgeAdvTrigSrc resets the More >> Advance >> Digital Edge >> Source property.
Get/Set/Reset DigEdge_AdvTrig_Edge

int32 __CFUNC DAQmxGetDigEdgeAdvTrigEdge(TaskHandle taskHandle,
    int32 *data);

int32 __CFUNC DAQmxSetDigEdgeAdvTrigEdge(TaskHandle taskHandle,
    int32 data);

int32 __CFUNC DAQmxResetDigEdgeAdvTrigEdge(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeAdvTrigEdge gets the property.
DAQmxSetDigEdgeAdvTrigEdge sets the property.
DAQmxResetDigEdgeAdvTrigEdge resets the property.
Get/Set/Reset DigEdge_AdvTrig_DigFltr_Enable

int32 __CFUNC DAQmxGetDigEdgeAdvTrigDigFltrEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigEdgeAdvTrigDigFltrEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetDigEdgeAdvTrigDigFltrEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeAdvTrigDigFltrEnable gets the More >> Advance >> Digital Edge >> Digital Filter >> Enable property.

DAQmxSetDigEdgeAdvTrigDigFltrEnable sets the More >> Advance >> Digital Edge >> Digital Filter >> Enable property.

DAQmxResetDigEdgeAdvTrigDigFltrEnable resets the More >> Advance >> Digital Edge >> Digital Filter >> Enable property.
Get/Set/Reset HshkTrig_Type

int32 __CFUNC DAQmxGetHshkTrigType(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetHshkTrigType(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetHshkTrigType(TaskHandle taskHandle);
Purpose

DAQmxGetHshkTrigType gets the More >> Handshake >> Trigger Type property.
DAQmxSetHshkTrigType sets the More >> Handshake >> Trigger Type property.
DAQmxResetHshkTrigType sets the More >> Handshake >> Trigger Type property.
Get/Set/Reset Interlocked_HshkTrig_Src

int32 __CFUNC DAQmxGetInterlockedHshkTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetInterlockedHshkTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetInterlockedHshkTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetInterlockedHshkTrigSrc gets the More >> Handshake >> Interlocked >> Source property.
DAQmxSetInterlockedHshkTrigSrc sets the More >> Handshake >> Interlocked >> Source property.
DAQmxResetInterlockedHshkTrigSrc resets the More >> Handshake >> Interlocked >> Source property.
Get/Set/Reset Interlocked_HshkTrig_AssertedLvl

```c
int32 __CFUNC DAQmxGetInterlockedHshkTrigAssertedLvl(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetInterlockedHshkTrigAssertedLvl(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetInterlockedHshkTrigAssertedLvl(TaskHandle taskHandle);
```
Purpose

DAQmxGetInterlockedHshkTrigAssertedLvl gets the Interlocked Asserted Level property.

DAQmxSetInterlockedHshkTrigAssertedLvl sets the Interlocked Asserted Level property.

DAQmxResetInterlockedHshkTrigAssertedLvl resets the Interlocked Asserted Level property.
Get/Set/Reset PauseTrig_Type

int32 __CFUNC DAQmxGetPauseTrigType(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetPauseTrigType(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetPauseTrigType(TaskHandle taskHandle);
Purpose

DAQmxGetPauseTrigType gets the More >> Pause >> Trigger Type property.

DAQmxSetPauseTrigType sets the More >> Pause >> Trigger Type property.

DAQmxResetPauseTrigType resets the More >> Pause >> Trigger Type property.
Get/Set/Reset AnlgLvl_PauseTrig_Src

int32 __CFUNC DAQmxGetAnlgLvlPauseTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAnlgLvlPauseTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAnlgLvlPauseTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgLvlPauseTrigSrc gets the More >> Pause >> Analog Level >> Source property.

DAQmxSetAnlgLvlPauseTrigSrc sets the More >> Pause >> Analog Level >> Source property.

DAQmxResetAnlgLvlPauseTrigSrc resets the More >> Pause >> Analog Level >> Source property.
Get/Set/Reset AnlgLvl_PauseTrig_When

int32 __CFUNC DAQmxGetAnlgLvlPauseTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgLvlPauseTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgLvlPauseTrigWhen(TaskHandle taskHandle);
**Purpose**

DAQmxGetAnlgLvlPauseTrigWhen gets the More >> Pause >> Analog Level >> Pause When property.

DAQmxSetAnlgLvlPauseTrigWhen sets the More >> Pause >> Analog Level >> Pause When property.

DAQmxResetAnlgLvlPauseTrigWhen resets the More >> Pause >> Analog Level >> Pause When property.
Get/Set/Reset AnlgLvl_PauseTrig_Lvl

int32 __CFUNC DAQmxGetAnlgLvlPauseTrigLvl(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgLvlPauseTrigLvl(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgLvlPauseTrigLvl(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgLvlPauseTrigLvl gets the More >> Pause >> Analog Level >> Level property.

DAQmxSetAnlgLvlPauseTrigLvl sets the More >> Pause >> Analog Level >> Level property.

DAQmxResetAnlgLvlPauseTrigLvl resets the More >> Pause >> Analog Level >> Level property.
Get/Set/Reset AnlgLvl_PauseTrig_Hyst

int32 __CFUNC DAQmxGetAnlgLvlPauseTrigHyst(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgLvlPauseTrigHyst(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgLvlPauseTrigHyst(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgLvlPauseTrigHyst gets the More >> Pause >> Analog Level >> Hysteresis property.

DAQmxSetAnlgLvlPauseTrigHyst sets the More >> Pause >> Analog Level >> Hysteresis property.

DAQmxResetAnlgLvlPauseTrigHyst resets the More >> Pause >> Analog Level >> Hysteresis property.
Get/Set/Reset AnlgLvl_PauseTrig_Coupling

int32 __CFUNC DAQmxGetAnlgLvlPauseTrigCoupling(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgLvlPauseTrigCoupling(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgLvlPauseTrigCoupling(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgLvlPauseTrigCoupling gets the property.
DAQmxSetAnlgLvlPauseTrigCoupling sets the property.
DAQmxResetAnlgLvlPauseTrigCoupling resets the property.
Get/Set/Reset AnlgWin_PauseTrig_Src

```c
int32 __CFUNC DAQmxGetAnlgWinPauseTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetAnlgWinPauseTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetAnlgWinPauseTrigSrc(TaskHandle taskHandle);
```
Purpose

DAQmxGetAnlgWinPauseTrigSrc gets the property.

DAQmxSetAnlgWinPauseTrigSrc sets the property.

DAQmxResetAnlgWinPauseTrigSrc resets the property.
Get/Set/Reset AnlgWinPauseTrigWhen

int32 __CFUNC DAQmxGetAnlgWinPauseTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgWinPauseTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgWinPauseTrigWhen(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinPauseTrigWhen gets the property.

DAQmxSetAnlgWinPauseTrigWhen sets the property.

DAQmxResetAnlgWinPauseTrigWhen resets the property.
Get/Set/Reset AnlgWin_PauseTrig_Top

int32 __CFUNC DAQmxGetAnlgWinPauseTrigTop(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgWinPauseTrigTop(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgWinPauseTrigTop(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinPauseTrigTop gets the property.

DAQmxSetAnlgWinPauseTrigTop sets the property.

DAQmxResetAnlgWinPauseTrigTop resets the property.
Get/Set/Reset AnlgWin_PauseTrig_Btm

int32 __CFUNC DAQmxGetAnlgWinPauseTrigBtm(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetAnlgWinPauseTrigBtm(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetAnlgWinPauseTrigBtm(TaskHandle taskHandle);
**Purpose**

DAQmxGetAnlgWinPauseTrigBtm gets the [More >> Pause >> Analog Window >> Bottom](#) property.

DAQmxSetAnlgWinPauseTrigBtm sets the [More >> Pause >> Analog Window >> Bottom](#) property.

DAQmxResetAnlgWinPauseTrigBtm resets the [More >> Pause >> Analog Window >> Bottom](#) property.
Get/Set/Reset AnlgWin_PauseTrig_Coupling

int32 __CFUNC DAQmxGetAnlgWinPauseTrigCoupling(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetAnlgWinPauseTrigCoupling(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetAnlgWinPauseTrigCoupling(TaskHandle taskHandle);
Purpose

DAQmxGetAnlgWinPauseTrigCoupling gets the property.

DAQmxSetAnlgWinPauseTrigCoupling sets the property.

DAQmxResetAnlgWinPauseTrigCoupling resets the property.
Get/Set/Reset DigLvl_PauseTrig_Src

int32 __CFUNC DAQmxGetDigLvlPauseTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigLvlPauseTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigLvlPauseTrigSrc(TaskHandle taskHandle);
Purpose

DAQmxGetDigLvlPauseTrigSrc gets the [More >> Pause >> Digital Level >> Source](http://example.com) property.

DAQmxSetDigLvlPauseTrigSrc sets the [More >> Pause >> Digital Level >> Source](http://example.com) property.

DAQmxResetDigLvlPauseTrigSrc resets the [More >> Pause >> Digital Level >> Source](http://example.com) property.
Get/Set/Reset DigLvl_PauseTrig_When

int32 __CFUNC DAQmxGetDigLvlPauseTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDigLvlPauseTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDigLvlPauseTrigWhen(TaskHandle taskHandle);
Purpose

DAQmxGetDigLvlPauseTrigWhen gets the More >> Pause >> Digital Level >> Pause When property.
DAQmxSetDigLvlPauseTrigWhen sets the More >> Pause >> Digital Level >> Pause When property.
DAQmxResetDigLvlPauseTrigWhen resets the More >> Pause >> Digital Level >> Pause When property.
Get/Set/Reset DigLvl_PauseTrig_DigFltr_Enable

int32 __CFUNC DAQmxGetDigLvlPauseTrigDigFltrEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigLvlPauseTrigDigFltrEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetDigLvlPauseTrigDigFltrEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigLvlPauseTrigDigFltrEnable gets the More >> Pause >> Digital Level >> Digital Filter >> Enable property.

DAQmxSetDigLvlPauseTrigDigFltrEnable sets the More >> Pause >> Digital Level >> Digital Filter >> Enable property.

DAQmxResetDigLvlPauseTrigDigFltrEnable resets the More >> Pause >> Digital Level >> Digital Filter >> Enable property.
Get/Set/Reset
DigLvl_PauseTrig_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetDigLvlPauseTrigDigFltrMinPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC
    DAQmxSetDigLvlPauseTrigDigFltrMinPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC
    DAQmxResetDigLvlPauseTrigDigFltrMinPulseWidth(TaskHandle taskHandle);
**Purpose**

DAQmxGetDigLvlPauseTrigDigFltrMinPulseWidth gets the property.

DAQmxSetDigLvlPauseTrigDigFltrMinPulseWidth sets the property.

DAQmxResetDigLvlPauseTrigDigFltrMinPulseWidth resets the property.
Get/Set/Reset
DigLvl_PauseTrig_DigFltr_TimebaseSrc

int32 __CFUNC DAQmxGetDigLvlPauseTrigDigFltrTimebaseSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigLvlPauseTrigDigFltrTimebaseSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetDigLvlPauseTrigDigFltrTimebaseSrc(TaskHandle taskHandle);
Purpose

DAQmxGetDigLvlPauseTrigDigFltrTimebaseSrc gets the Digital Filter >> Timebase >> Source property.

DAQmxSetDigLvlPauseTrigDigFltrTimebaseSrc sets the Digital Filter >> Timebase >> Source property.

DAQmxResetDigLvlPauseTrigDigFltrTimebaseSrc resets the Digital Filter >> Timebase >> Source property.
Get/Set/Reset

DigLvl_PauseTrig_DigFltr_TimebaseRate

int32 __CFUNC DAQmxGetDigLvlPauseTrigDigFltrTimebaseRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetDigLvlPauseTrigDigFltrTimebaseRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC
    DAQmxResetDigLvlPauseTrigDigFltrTimebaseRate(TaskHandle taskHandle);
Purpose

DAQmxGetDigLvlPauseTrigDigFltrTimebaseRate gets the property.

DAQmxSetDigLvlPauseTrigDigFltrTimebaseRate sets the property.

DAQmxResetDigLvlPauseTrigDigFltrTimebaseRate resets the property.
Get/Set/Reset DigLvl_PauseTrig_DigSync_Enable

int32 __CFUNC DAQmxGetDigLvlPauseTrigDigSyncEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigLvlPauseTrigDigSyncEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetDigLvlPauseTrigDigSyncEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigLvlPauseTrigDigSyncEnable gets the property.

DAQmxSetDigLvlPauseTrigDigSyncEnable sets the property.

DAQmxResetDigLvlPauseTrigDigSyncEnable resets the property.
Get/Set/Reset DigPattern_PauseTrig_Src

int32 __CFUNC DAQmxGetDigPatternPauseTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigPatternPauseTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigPatternPauseTrigSrc(TaskHandle taskHandle);
Purpose

Get/Set/Reset DigPattern_PauseTrig_Pattern

int32 __CFUNC DAQmxGetDigPatternPauseTrigPattern(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigPatternPauseTrigPattern(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigPatternPauseTrigPattern(TaskHandle taskHandle);
**Purpose**

DAQmxGetDigPatternPauseTrigPattern gets the property.

DAQmxSetDigPatternPauseTrigPattern sets the property.

DAQmxResetDigPatternPauseTrigPattern resets the property.
Get/Set/Reset DigPattern_PauseTrig_When

int32 __CFUNC DAQmxGetDigPatternPauseTrigWhen(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDigPatternPauseTrigWhen(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDigPatternPauseTrigWhen(TaskHandle taskHandle);
Purpose

DAQmxGetDigPatternPauseTrigWhen gets the property.

DAQmxSetDigPatternPauseTrigWhen sets the property.

DAQmxResetDigPatternPauseTrigWhen resets the property.
Get/Set/Reset ArmStartTrig_Type

int32 __CFUNC DAQmxGetArmStartTrigType(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetArmStartTrigType(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetArmStartTrigType(TaskHandle taskHandle);
Purpose

DAQmxGetArmStartTrigType gets the Arm Start Trigger property.
DAQmxSetArmStartTrigType sets the Arm Start Trigger property.
DAQmxResetArmStartTrigType resets the Arm Start Trigger property.
Get/Set/Reset DigEdge_ArmStartTrig_Src

int32 __CFUNC DAQmxGetDigEdgeArmStartTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigEdgeArmStartTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigEdgeArmStartTrig Src(TaskHandle taskHandle);
**Purpose**

DAQmxGetDigEdgeArmStartTrigSrc gets the property.

DAQmxSetDigEdgeArmStartTrigSrc sets the property.

DAQmxResetDigEdgeArmStartTrigSrc resets the property.
Get/Set/Reset DigEdge_ArmStartTrig_Edge

int32 __CFUNC DAQmxGetDigEdgeArmStartTrigEdge(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDigEdgeArmStartTrigEdge(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDigEdgeArmStartTrigEdge(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeArmStartTrigEdge gets the property.

DAQmxSetDigEdgeArmStartTrigEdge sets the property.

DAQmxResetDigEdgeArmStartTrigEdge resets the property.
Get/Set/Reset
DigEdge_ArmStartTrig_DigFltr_Enable

int32 __CFUNC DAQmxGetDigEdgeArmStartTrigDigFltrEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigEdgeArmStartTrigDigFltrEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetDigEdgeArmStartTrigDigFltrEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeArmStartTrigDigFltrEnable gets the property.

DAQmxSetDigEdgeArmStartTrigDigFltrEnable sets the property.

DAQmxResetDigEdgeArmStartTrigDigFltrEnable resets the property.
Get/Set/Reset
DigEdge_ArmStartTrig_DigFltr_MinPulseWidth

int32 __CFUNC
    DAQmxGetDigEdgeArmStartTrigDigFltrMinPulseWidth(TaskHandle taskHandle, float64 *data);

int32 __CFUNC
    DAQmxSetDigEdgeArmStartTrigDigFltrMinPulseWidth(TaskHandle taskHandle, float64 data);

int32 __CFUNC
    DAQmxResetDigEdgeArmStartTrigDigFltrMinPulseWidth(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeArmStartTrigDigFltrMinPulseWidth gets the property.

DAQmxSetDigEdgeArmStartTrigDigFltrMinPulseWidth sets the property.

DAQmxResetDigEdgeArmStartTrigDigFltrMinPulseWidth resets the property.
Get/Set/Reset
DigEdge_ArmStartTrig_DigFltr_TimebaseSrc

int32 __CFUNC
    DAQmxGetDigEdgeArmStartTrigDigFltrTimebaseSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC
    DAQmxSetDigEdgeArmStartTrigDigFltrTimebaseSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC
    DAQmxResetDigEdgeArmStartTrigDigFltrTimebaseSrc(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeArmStartTrigDigFltrTimebaseSrc gets the property.

DAQmxSetDigEdgeArmStartTrigDigFltrTimebaseSrc sets the property.

DAQmxResetDigEdgeArmStartTrigDigFltrTimebaseSrc resets the property.
Get/Set/Reset
DigEdge_ArmStartTrig_DigFltr_TimebaseRate

int32 __CFUNC
   DAQmxGetDigEdgeArmStartTrigDigFltrTimebaseRate(TaskHandle taskHandle, float64 *data);

int32 __CFUNC
   DAQmxSetDigEdgeArmStartTrigDigFltrTimebaseRate(TaskHandle taskHandle, float64 data);

int32 __CFUNC
   DAQmxResetDigEdgeArmStartTrigDigFltrTimebaseRate(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeArmStartTrigDigFltrTimebaseRate gets the Digital Edge >> Digital Filter >> Timebase >> Rate property.

DAQmxSetDigEdgeArmStartTrigDigFltrTimebaseRate sets the Digital Edge >> Digital Filter >> Timebase >> Rate property.

DAQmxResetDigEdgeArmStartTrigDigFltrTimebaseRate resets the Digital Edge >> Digital Filter >> Timebase >> Rate property.
Get/Set/Reset
DigEdge_ArmStartTrig_DigSync_Enable

int32 __CFUNC DAQmxGetDigEdgeArmStartTrigDigSyncEnable(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetDigEdgeArmStartTrigDigSyncEnable(TaskHandle taskHandle, bool32 data);

int32 __CFUNC
    DAQmxResetDigEdgeArmStartTrigDigSyncEnable(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeArmStartTrigDigSyncEnable gets the Digital Synchronization >> Enable property.

DAQmxSetDigEdgeArmStartTrigDigSyncEnable sets the Digital Synchronization >> Enable property.

DAQmxResetDigEdgeArmStartTrigDigSyncEnable resets the Digital Synchronization >> Enable property.
Get/Set/Reset Watchdog Timeout

int32 __CFUNC DAQmxGetWatchdogTimeout(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetWatchdogTimeout(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetWatchdogTimeout(TaskHandle taskHandle);
Purpose

DAQmxGetWatchdogTimeout gets the Timeout property.
DAQmxSetWatchdogTimeout sets the Timeout property.
DAQmxResetWatchdogTimeout resets the Timeout property.
Get/Set/Reset WatchdogExpireTrig_Type

int32 __CFUNC DAQmxGetWatchdogExpireTrigType(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetWatchdogExpireTrigType(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetWatchdogExpireTrigType(TaskHandle taskHandle);
Purpose

DAQmxGetWatchdogExpirTrigType gets the \texttt{Expiration\ Trigger >> Trigger\ Type} property.

DAQmxSetWatchdogExpirTrigType sets the \texttt{Expiration\ Trigger >> Trigger\ Type} property.

DAQmxResetWatchdogExpirTrigType resets the \texttt{Expiration\ Trigger >> Trigger\ Type} property.
Get/Set/Reset DigEdge_WatchdogExpirTrig_Src

```c
int32 __CFUNC DAQmxGetDigEdgeWatchdogExpirTrigSrc(TaskHandle taskHandle, char *data, uInt32 bufferSize);

int32 __CFUNC DAQmxSetDigEdgeWatchdogExpirTrigSrc(TaskHandle taskHandle, const char *data);

int32 __CFUNC DAQmxResetDigEdgeWatchdogExpirTrigSrc(TaskHandle taskHandle);
```
Purpose

DAQmxGetDigEdgeWatchdogExpirTrigSrc gets the Expiration Trigger >> Digital Edge >> Source property.

DAQmxSetDigEdgeWatchdogExpirTrigSrc sets the Expiration Trigger >> Digital Edge >> Source property.

DAQmxResetDigEdgeWatchdogExpirTrigSrc resets the Source property.
Get/Set/Reset DigEdge_WatchdogExpireTrig_Edge

int32 __CFUNC DAQmxGetDigEdgeWatchdogExpireTrigEdge(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetDigEdgeWatchdogExpireTrigEdge(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetDigEdgeWatchdogExpireTrigEdge(TaskHandle taskHandle);
Purpose

DAQmxGetDigEdgeWatchdogExpIrTrigEdge gets the Expiration Trigger >> Digital Edge >> Edge property.

DAQmxSetDigEdgeWatchdogExpIrTrigEdge sets the Expiration Trigger >> Digital Edge >> Edge property.

DAQmxResetDigEdgeWatchdogExpIrTrigEdge resets the Edge property.
Get/Set/Reset Watchdog_DO_ExpirState

int32 __CFUNC DAQmxGetWatchdogDOExpirState(TaskHandle taskHandle, const char lines[], int32 *data);

int32 __CFUNC DAQmxSetWatchdogDOExpirState(TaskHandle taskHandle, const char lines[], int32 data);

int32 __CFUNC DAQmxResetWatchdogDOExpirState(TaskHandle taskHandle, const char lines[]);
Purpose

DAQmxGetWatchdogDOExpireState gets the 
Expiration States >> Digital Output >> Expiration State
property.

DAQmxSetWatchdogDOExpireState sets the 
Expiration States >> Digital Output >> Expiration State
property.

DAQmxResetWatchdogDOExpireState resets the 
Expiration States >> Digital Output >> Expiration State
property.
Get/Set/Reset Watchdog_HasExpired

int32 __CFUNC DAQmxGetWatchdogHasExpired(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetWatchdogHasExpired gets the Status >> Expired property.
Get/Set/Reset WriteRelativeTo

int32 __CFUNC DAQmxGetWriteRelativeTo(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetWriteRelativeTo(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetWriteRelativeTo(TaskHandle taskHandle);
Purpose

DAQmxGetWriteRelativeTo gets the Relative To property.
DAQmxSetWriteRelativeTo sets the Relative To property.
DAQmxResetWriteRelativeTo resets the Relative To property.
**Get/Set/Reset Write_Offset**

int32 __CFUNC DAQmxGetWriteOffset(TaskHandle taskHandle, int32 *data);
int32 __CFUNC DAQmxSetWriteOffset(TaskHandle taskHandle, int32 data);
int32 __CFUNC DAQmxResetWriteOffset(TaskHandle taskHandle);
Purpose

DAQmxGetWriteOffset gets the Offset property.
DAQmxSetWriteOffset sets the Offset property.
DAQmxResetWriteOffset resets the Offset property.
Get/Set/Reset Write_RegenMode

int32 __CFUNC DAQmxGetWriteRegenMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetWriteRegenMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetWriteRegenMode(TaskHandle taskHandle);
Purpose

DAQmxGetWriteRegenMode gets the Regeneration Mode property.
DAQmxSetWriteRegenMode sets the Regeneration Mode property.
DAQmxResetWriteRegenMode resets the Regeneration Mode property.
Get/Set/Reset Write_CurrWritePos

int32 __CFUNC DAQmxGetWriteCurrWritePos(TaskHandle taskHandle, uInt64 *data);
Purpose

DAQmxGetWriteCurrWritePos gets the `Status >> Current Write Position` property.
Get/Set/Reset Write_OvercurrentChansExist

int32 __CFUNC DAQmxGetWriteOvercurrentChansExist(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetWriteOvercurrentChansExist gets the Status >> Overcurrent >> Overcurrent Channels Exist property.
Get/Set/Reset Write_OvercurrentChans

int32 __CFUNC DAQmxGetWriteOvercurrentChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);
Purpose

DAQmxGetWriteOvercurrentChans gets the `Status >> Overcurrent >> Overcurrent Channels` property.
Get/Set/Reset Write_OpenCurrentLoopChansExist

int32 __CFUNC DAQmxGetWriteOpenCurrentLoopChansExist(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetWriteOpenCurrentLoopChansExist gets the Status >> Open Current Loop >> Open Current Loop Channels Exist property.
Get/Set/Reset Write_OpenCurrentLoopChans

int32 __CFUNC DAQmxGetWriteOpenCurrentLoopChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);
Purpose

DAQmxGetWriteOpenCurrentLoopChans gets the `Status >> Open Current Loop >> Open Current Loop Channels` property.
Get/Set/Reset Write_PowerSupplyFaultChansExist

int32 __CFUNC DAQmxGetWritePowerSupplyFaultChansExist(TaskHandle taskHandle, bool32 *data);
Purpose

DAQmxGetWritePowerSupplyFaultChansExist gets the \texttt{Status >> Power Supply Fault >> Power Supply Fault Channels Exist} property.
Get/Set/Reset Write_PowerSupplyFaultChans

int32 __CFUNC DAQmxGetWritePowerSupplyFaultChans(TaskHandle taskHandle, char *data, uInt32 bufferSize);
Purpose

Get/Set/Reset Write_SpaceAvail

int32 __CFUNC DAQmxGetWriteSpaceAvail(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetWriteSpaceAvail gets the Status >> Space Available in Buffer property.
Get/Set/Reset Write_TotalSampPerChanGenerated

int32 __CFUNC DAQmxGetWriteTotalSampPerChanGenerated(TaskHandle taskHandle, uInt64 *data);
Purpose

DAQmxGetWriteTotalSampPerChanGenerated gets the Status >> Total Samples Per Channel Generated property.
Get/Set/Reset Write_RawDataWidth

int32 __CFUNC DAQmxGetWriteRawDataWidth(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetWriteRawDataWidth gets the Advanced >> Raw Data Width property.
Get/Set/Reset Write_NumChans

int32 __CFUNC DAQmxGetWriteNumChans(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetWriteNumChans gets the Advanced >> Number of Channels property.
Get/Set/Reset Write_WaitMode

int32 __CFUNC DAQmxGetWriteWaitMode(TaskHandle taskHandle, int32 *data);

int32 __CFUNC DAQmxSetWriteWaitMode(TaskHandle taskHandle, int32 data);

int32 __CFUNC DAQmxResetWriteWaitMode(TaskHandle taskHandle);
**Purpose**

DAQmxGetWriteWaitMode gets the Advanced >> Wait Mode property.

DAQmxSetWriteWaitMode sets the Advanced >> Wait Mode property.

DAQmxResetWriteWaitMode resets the Advanced >> Wait Mode property.
Get/Set/Reset Write_SleepTime

int32 __CFUNC DAQmxGetWriteSleepTime(TaskHandle taskHandle, float64 *data);

int32 __CFUNC DAQmxSetWriteSleepTime(TaskHandle taskHandle, float64 data);

int32 __CFUNC DAQmxResetWriteSleepTime(TaskHandle taskHandle);
Purpose

DAQmxGetWriteSleepTime gets the Advanced >> Sleep Time property.
DAQmxSetWriteSleepTime sets the Advanced >> Sleep Time property.
DAQmxResetWriteSleepTime resets the Advanced >> Sleep Time property.
**Get/Set/Reset Write\_NextWriteIsLast**

```c
int32 __CFUNC DAQmxGetWriteNextWriteIsLast(TaskHandle taskHandle, bool32 *data);

int32 __CFUNC DAQmxSetWriteNextWriteIsLast(TaskHandle taskHandle, bool32 data);

int32 __CFUNC DAQmxResetWriteNextWriteIsLast(TaskHandle taskHandle);
```
Purpose

DAQmxGetWriteNextWriteIsLast gets the Advanced >> Next Write Is Last property.

DAQmxSetWriteNextWriteIsLast sets the Advanced >> Next Write Is Last property.

DAQmxResetWriteNextWriteIsLast resets the Advanced >> Next Write Is Last property.
Get/Set/Reset Write_DigitalLines_BytesPerChan

int32 __CFUNC DAQmxGetWriteDigitalLinesBytesPerChan(TaskHandle taskHandle, uInt32 *data);
Purpose

DAQmxGetWriteDigitalLinesBytesPerChan gets the Advanced >> Digital Output >> Number of Booleans Per Channel property.