# DAQmxAddGlobalChansToTask

int32 DAQmxAddGlobalChansToTask (TaskHandle taskHandle, const char channelNames[]);

Adds global virtual channels from MAX to the given task.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to which to add the channels from MAX.
channelNames	const char []	The channels to add to the task. You can <u>specify a list or range of channels</u> . These channels must be valid channels available from MAX. If you pass an invalid channel, NI-DAQmx returns an error. This value is ignored if it is empty.

#### Name Type Description

# DAQmxClearTask

int32 DAQmxClearTask (TaskHandle taskHandle);

Clears the task. Before clearing, this function stops the task, if necessary, and releases any resources reserved by the task. You cannot use a task once you clear the task without recreating or reloading the task.

If you use the DAQmxCreateTask function or any of the NI-DAQmx Create Channel functions within a loop, use this function within the loop after you finish with the task to avoid allocating unnecessary memory.

InputNameTypeDescriptiontaskHandleTaskHandleThe task to clear.

#### Name Type Description

# DAQmxCreateTask

int32 DAQmxCreateTask (const char taskName[], TaskHandle \*taskHandle);

Creates a  $_{\underline{task}}$  . If you use this function to create a task, you must use DAQmxClearTask to destroy it.

If you use this function within a loop, NI-DAQmx creates a new task in each iteration of the loop. Use the DAQmxClearTask function within the loop after you finish with the task to avoid allocating unnecessary memory.

Input

Name Type Description

**taskName** const char [] Name assigned to the task.

 $\overline{\mathbb{N}}$ 

**Note** This name may be changed internally. If you are using the C API, call DAQmxGetTaskName to verify whether the name was changed during creation. If you are using the CVI API, call DAQmxGetTaskAttribute with attribute ID DAQmx\_Task\_Name to verify the name change.

Output

Name Type Description

taskHandle TaskHandle \* A reference to the task created in this function.

#### Name Type Description

# DAQmxIsTaskDone

int32 DAQmxIsTaskDone (TaskHandle taskHandle, bool32 \*isTaskDone);

Queries the status of the task and indicates if it completed execution. Use this function to <u>ensure that the specified operation is complete</u> before you stop the task.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
Output		
Name	Туре	Description
isTaskDone	bool32 *	Indicates whether the measurement or generation completed.

#### Name Type Description

# DAQmxLoadTask

int32 DAQmxLoadTask (const char taskName[], TaskHandle \*taskHandle);

Loads an existing named <sub>task</sub> from Measurement & Automation Explorer (MAX). If you use this function to load a task, you must use DAQmxClearTask to destroy it.

Input		
Name	Туре	Description
taskName	const char []	A named task in MAX.
Output		
Name	Туре	Description
taskHandle	TaskHandle *	A reference to the task returned by this function.

#### Name Type Description

# DAQmxStartTask

int32 DAQmxStartTask (TaskHandle taskHandle);

Transitions the task from the committed state to the running state, which begins measurement or generation. Using this function is required for some applications and optional for others.

If you do not use this function, a measurement task starts automatically when a read operation begins. The **autoStart** parameter of the NI-DAQmx Write functions determines if a generation task starts automatically when you use an NI-DAQmx Write function.

If you do not call DAQmxStartTask and DAQmxStopTask when you call NI-DAQmx Read functions or NI-DAQmx Write functions multiple times, such as in a loop, the task starts and stops repeatedly. Starting and stopping a task repeatedly reduces the performance of the application.

InputNameTypeDescriptiontaskHandleTaskHandleThe task to start.

#### Name Type Description

# DAQmxStopTask

int32 DAQmxStopTask (TaskHandle taskHandle);

Stops the task and returns it to the state it was in before you called DAQmxStartTask or called an NI-DAQmx Write function with **autoStart** set to TRUE.

If you do not call DAQmxStartTask and DAQmxStopTask when you call NI-DAQmx Read functions or NI-DAQmx Write functions multiple times, such as in a loop, the task starts and stops repeatedly. Starting and stopping a task repeatedly reduces the performance of the application.

InputTypeDescriptiontaskHandleTaskHandleThe task to stop.

#### Name Type Description

# DAQmxRegisterDoneEvent

int32 DAQmxRegisterDoneEvent (TaskHandle taskHandle, uInt32 options, DAQmxDoneEventCallbackPtr callbackFunction, void \*callbackData);

Registers a callback function to receive an event when a task stops due to an error or when a finite acquisition task or finite generation task completes execution. A Done event does not occur when a task is stopped explicitly, such as by calling <sub>DAQmxStopTask</sub>.

Input Name

taskHandle

options

TaskHandle uInt32

Туре

Description

The task used in this function.

Use this parameter to set certain options. You can combine fla operator ('|') to set multiple options. Pass a value of zero if no

#### Value

0

DAQmx\_Val\_SynchronousEventCallback



🔊 Note If you are receiving synchronou than you are processing them, then th of your application might become uni

callbackFunction DAQmxDoneEventCallbackPtr The function that you want DAQmx to call when the event oc pass in this parameter must have the following prototype: int32 CVICALLBACK Callback (TaskHandle taskHandle, in

		*callbackData);
		Upon entry to the callback, the <b>taskHandle</b> parameter contain on which the event occurred. The status parameter contains th when the event occurred. If the status value is negative, it ind status value is zero, it indicates no error. If the status value is warning. The <b>callbackData</b> parameter contains the value you <b>callbackData</b> parameter of this function.
callbackData	void *	A value that you want DAQmx to pass to the callback functio parameter. Do not pass the address of a local variable or any ( not be valid when the function is executed.

#### Name Type Description

# DAQmxRegisterEveryNSamplesEvent

int32 DAQmxRegisterEveryNSamplesEvent (TaskHandle taskHandle, int32 everyNsamplesEventType, uInt32 nSamples, uInt32 options, DAQmxEveryNSamplesEventCallbackPtr callbackFunction, void \*callbackData);

Registers a callback function to receive an event when the specified number of samples is written from the device to the buffer or from the buffer to the device. This function only works with devices that support buffered tasks. When you stop a task explicitly any pending events are discarded. For example, if you call <u>DAQmxStopTask</u> then you do not receive any pending events.

Input

 Name
 Type

 taskHandle
 TaskHandle

 everyNsamplesEventType
 int32

**Description** The task used in this function. The type of event you want to receive.

Value DAQmx\_Val\_Acquired\_Int

DAQmx\_Val\_Transferred\_I
nSamplesuInt32The number of samples after which eachoptionsuInt32Use this parameter to set certain options.<br/>operator ('|') to set multiple options. Pass

#### Value

0

DAQmx\_Val\_Synchronous



callbackData

void \*

A value that you want DAQmx to pass to parameter. Do not pass the address of a lo not be valid when the function is executed

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxRegisterSignalEvent

Registers a callback function to receive an event when the specified  $_{\underline{hardware\ event}}$  occurs.

When you stop a task explicitly any pending events are discarded. For example, if you call <u>DAQmxStopTask</u> then you do not receive any pending events.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
signalID	int32	The signal for which you want to receive results.
		Value
		DAQmx_Val_SampleClock
		DAQmx_Val_SampleCompleteEvent
		DAQmx_Val_ChangeDetectionEvent
		DAQmx_Val_CounterOutputEvent
options	uInt32	Use this parameter to set certain options. You can combine for operator (' ') to set multiple options. Pass a value of zero if no
		Value
		0

DAQmx\_Val\_SynchronousEventCallbac

		<b>Note</b> If you are receiving synchronou than you are processing them, then the of your application might become un
callbackFunction	DAQmxSignalEventCallbackPtr	The function that you want DAQmx to call when the event c pass in this parameter must have the following prototype: int32 CVICALLBACK Callback (TaskHandle taskHandle, i *callbackData); Upon entry to the callback, the <b>taskHandle</b> parameter conta on which the event occurred. The <b>signalID</b> parameter contai in the <b>signalID</b> parameter of this function. The <b>callbackDat</b> value you passed in the <b>callbackData</b> parameter of this func
callbackData	void *	A value that you want DAQmx to pass to the callback functi parameter. Do not pass the address of a local variable or any not be valid when the function is executed.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxGetNthTaskChannel

int32 DAQmxGetNthTaskChannel (TaskHandle taskHandle, uInt32 index, char buffer[], int32 bufferSize);

Returns the *N*th channel. This function takes the **taskHandle**, **index**, and **bufferSize** you specify and returns the *N*th channel.

Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
index	uInt32	The <i>N</i> th channel you want to return. The <b>index</b> starts at 1.	
bufferSize	int32	The size, in bytes, of <b>buffer</b> . If you pass 0, this function returns the number of byt needed to allocate.	
Output			
buffer	char []	The <i>N</i> th channel in the index. If you pass NULL, this function returns the number of bytes needed to allocate.	

#### Name Type Description

status int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error. For this function, if you pass NULL for the buffer or 0 for the buffer size, this function returns the number of bytes needed to allocate.

# DAQmxGetNthTaskDevice

int32 DAQmxGetNthTaskDevice (TaskHandle taskHandle, uInt32 index, char buffer[], int32 bufferSize);

Returns the *N*th device. This function takes the **taskHandle**, **index**, and **bufferSize** you specify and returns the *N*th device.

Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
index	uInt32	The <i>N</i> th device you want to return. The <b>index</b> starts at 1.	
bufferSize	int32	The size, in bytes, of <b>buffer</b> . If you pass 0, this function returns the number of bytes needed to allocate.	
Output			
buffer	char []	The <i>N</i> th device in the index. If you pass NULL, this function returns the number of bytes needed to allocate.	

#### Name Type Description

status int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error. For this function, if you pass NULL for the buffer or 0 for the buffer size, this function returns the number of bytes needed to allocate.

# DAQmxTaskControl

int32 DAQmxTaskControl (TaskHandle taskHandle, int32 action);

Alters the <sub>state</sub> of a task according to the **action** you specify. To minimize the time required to start a task, for example, DAQmxTaskControl can commit the task prior to starting.

Input

Name	Туре	Description					
taskHandle	TaskHandle	The task used in this function.					
action	int32	Specifies how to alter the task state.	Specifies how to alter the task state.				
		Value	Description				
		DAQmx_Val_Task_Start	Starts execution of the task.				
		DAQmx_Val_Task_Stop	Stops execution of the task.				
		DAQmx_Val_Task_Verify	Verifies that all task parameters are valid for the hardware.				
		DAQmx_Val_Task_Commit	Programs the hardware as much as possible according to the task configuration.				
		DAQmx_Val_Task_Reserve	Reserves the hardware resources needed for the task. No other tasks can reserve these same resources.				
		DAQmx_Val_Task_Unreserve	Releases all previously reserved resources.				
		DAQmx_Val_Task_Abort	Abort is used to stop an operation, such as Read or Write, that is currently active. Abort puts the task into an unstable but recoverable state. To recover the task, call Start to restart the task				

or call Stop to reset the task without starting it.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCreateAIAccelChan

int32 DAQmxCreateAIAccelChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, float64 sensitivity, int32 sensitivityUnits, int32 currentExcitSource, float64 currentExcitVal, const char customScaleName[]);

Creates channel(s) that use an *accelerometer* to measure acceleration and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
terminalConfig	int32	The $\underline{input\ terminal\ configuration}$ for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At DA	run time, NI- Qmx chooses
			defaul the	<u>lt terminal configuration</u> f channel.
		DAQmx_Val_RSE	<u>Refer</u>	enced single-ended mode
		DAQmx_Val_NRSE	Nonreferenced single-ended mo	
		DAQmx_Val_Diff		rential mode
		DAQmx_Val_PseudoDiff	<u>Pseud</u>	lodifferential mode
minVal	float64	The minimum value, in <b>units</b> , that you expect to	meas	ure.
maxVal	float64	The maximum value, in <b>units</b> , that you expect to	meas	sure.
units	int32	The units to use to return acceleration measure	emen	ts from the channel.
		Name		Description
		DAQmx_Val_AccelUnit_g		G. 1 g is approximately equal to 9.81 1
		DAQmx_Val_FromCustomSca	le	Units specifie a custom scale Use <b>customScale</b> to specify a cu

float64 The sensitivity of the sensor. This value is in the units you specify with sensitivity sensitivityUnits. Refer to the sensor documentation to determine this va sensitivityUnits int32 The units of **sensitivity**. Description Name DAQmx\_Val\_mVoltsPerG mVolts/g. DAQmx\_Val\_VoltsPerG Volts/g. The source of excitation. currentExcitSource int32 Value Description DAQmx Val Internal Use the built-in excita source of the device. must use **currentExc** to specify the amount excitation. DAQmx\_Val\_External Use an excitation sou other than the built-in excitation source of th device. You must use **currentExcitVal** to specify the amount of excitation. DAQmx\_Val\_None Supply no excitation 1 channel. You cannot ı this value if the senso requires excitation. currentExcitVal float64 The amount of excitation, in amperes, that the sensor requires. const char [] The name of a *custom scale* to apply to the channel. To use this parameter, customScaleName must set units to DAQmx\_Val\_FromCustomScale. If you do not set uni

DAQmx\_Val\_FromCustomScale, you must set customScaleName to N

scale.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCreateAICurrentChan

int32 DAQmxCreateAICurrentChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, int32 shuntResistorLoc, float64 extShuntResistorVal, const char customScaleName[]);

Creates channel(s) for current measurement and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
terminalConfig	int32	The input terminal configuration for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At NI- cho defau config cha	run time, -DAQmx ooses the <sup>It terminal</sup> guration for the annel.
		DAQmx_Val_RSE	<u>Refer</u> mode	enced single-ended
		DAQmx_Val_NRSE	<u>Nonre</u> mode	eferenced single-ended
		DAQmx_Val_Diff	<u>Diffe</u>	rential mode
		DAQmx_Val_PseudoDiff	<u>Pseud</u>	lodifferential mode
minVal maxVal units	float64 float64 int32	The <u>minimum value</u> , in <b>units</b> , that you expect to The <u>maximum value</u> , in <b>units</b> , that you expect to The units to use to return the measurement.	meas meas	ure. sure. Description
				Description
		DAQmx_val_Amps		amperes
		DAQmx_Val_FromCustomSca	ıle	Units a <u>custom</u> specifies. If you select this value,

you must specify a custom scale name.

shuntResistorLoc	int32	The location of the shunt resistor.	
		Value	Description
		DAQmx_Val_Default	At run time, NI- DAQmx chooses the default shunt resistor location for the channel.
		DAQmx_Val_Internal	Use the built-in shunt resistor of the device.
		DAQmx_Val_External	Use a shunt resistor external to the device. You must specify the value of the shunt resistor in <b>extShuntResistorVal</b> .
extShuntResistorVal	float64	The value, in ohms, of an external s	hunt resistor.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to	o the channel. To use this

bist char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, you must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **units** to DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to NULL.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCreateAICurrentRMSChan

int32 DAQmxCreateAICurrentRMSChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, int32 shuntResistorLoc, float64 extShuntResistorVal, const char customScaleName[]);

Creates channel(s) for RMS current measurement and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
terminalConfig	int32	The $\underline{input terminal configuration}$ for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At NI- cho defau config cha	run time, -DAQmx ooses the <sup>It terminal</sup> guration for the annel.
		DAQmx_Val_RSE	<u>Refer</u> mode	enced single-ended
		DAQmx_Val_NRSE	<u>Nonre</u> mode	eferenced single-ended
		DAQmx_Val_Diff	<u>Diffe</u>	rential mode
		DAQmx_Val_PseudoDiff	<u>Pseud</u>	lodifferential mode
minVal maxVal units	float64 float64 int32	The <u>minimum value</u> , in <b>units</b> , that you expect to The <u>maximum value</u> , in <b>units</b> , that you expect to The units to use to return the measurement.	meas meas	ure. sure. Description
				Description
		DAQmx_val_Amps		amperes
		DAQmx_Val_FromCustomSca	ıle	Units a <u>custom</u> specifies. If you select this value,

you must specify a custom scale name.

shuntResistorLoc	int32	The location of the shunt resistor.	
		Value	Description
		DAQmx_Val_Default	At run time, NI- DAQmx chooses the default shunt resistor location for the channel.
		DAQmx_Val_Internal	Use the built-in shunt resistor of the device.
		DAQmx_Val_External	Use a shunt resistor external to the device. You must specify the value of the shunt resistor in <b>extShuntResistorVal</b> .
extShuntResistorVal	float64	The value, in ohms, of an external s	hunt resistor.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to	o the channel. To use this

bist char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, you must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **units** to DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to NULL.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.
# DAQmxCreateAIFreqVoltageChan

int32 DAQmxCreateAIFreqVoltageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, float64 thresholdLevel, float64 hysteresis, const char customScaleName[]);

Creates channel(s) that use a frequency-to-voltage converter to measure frequency and adds the channel(s) to the task you specify with **taskHandle**.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you expect to measure.		
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to measure.		
units	int32	The units to use to return the measurement.		
		Name Description		
		DAQmx_Val_Hz hertz		
		DAQmx_Val_FromCustomScale	Units specifie a custom scale Use <b>customScale</b> to specify a cu scale.	
thresholdLevel	float64	The voltage level at which to recognize waveform	repetitions.	
hysteresis	float64	Specifies in volts a window below <b>thresholdLevel</b> . The input voltage n below <b>thresholdLevel</b> minus <b>hysteresis</b> before NI-DAQmx recognizes waveform repetition. Hysteresis can improve measurement accuracy wh signal contains noise or jitter.		
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set <b>units</b> to DAQmx_Val_FromCustomScale. If you do not set <b>uni</b> DAQmx_Val_FromCustomScale, you must set <b>customScaleName</b> to N		

#### Name Type Description

# DAQmxCreateAIMicrophoneChan

int32 DAQmxCreateAIMicrophoneChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, int32 units, float64 micSensitivity, float64 maxSndPressLevel, int32 currentExcitSource, float64 currentExcitVal, const char customScaleName[]);

Creates channel(s) that use a microphone to  $_{\frac{\text{measure sound pressure}}{\text{measure sound pressure}}}$  and adds the channel(s) to the  $_{\frac{\text{task}}{\text{sourd specify with taskHandle}}$ .

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
terminalConfig	int32	The input terminal configuration for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At DA	run time, NI- Qmx chooses
			<sub>defau</sub> the	<u>lt terminal configuration</u> f
		DAQmx_Val_RSE	<u>Refer</u>	renced single-ended mode
		DAQmx_Val_NRSE	Nonre	eferenced single-ended mo
		DAQmx_Val_Diff	<u>Diffe</u>	rential mode
		DAQmx_Val_PseudoDiff	Pseuc	lodifferential mode
units	int32	The units to use to return sound pressure mea	surem	ients.
		Name		Description
		DAQmx_Val_Pascals		pascals
		DAQmx_Val_FromCustomSca	ale	Units specifie a custom scale Use <b>customScale</b> to specify a cu scale.
micSensitivity	float64	The sensitivity of the microphone. Specify the	is valı	ue in millivolts per pa
maxSndPressLevel	float64	The maximum instantaneous sound pressure level you expect to measur value is in decibels, referenced to 20 micropascals.		

currentExcitSource	
--------------------	--

currentExcitVal customScaleName int32

The source of excitation.

	<b>.</b>		
	Value	Description	
	DAQmx_Val_Internal	Use the built-in excita source of the device. must use <b>currentExc</b> to specify the amount excitation.	
	DAQmx_Val_External	Use an excitation sour other than the built-in excitation source of th device. You must use <b>currentExcitVal</b> to specify the amount of excitation.	
	DAQmx_Val_None	Supply no excitation the channel. You cannot up this value if the senso requires excitation.	
float64	The amount of excitation, in amperes,	that the sensor requires.	
const char []	The name of a custom scale to apply to the	he channel. To use this parameter.	

onst char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **un**i DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to N

#### Name Type Description

# DAQmxCreateAIResistanceChan

int32 DAQmxCreateAIResistanceChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 currentExcitSource, float64 currentExcitVal, const char customScaleName[]);

Create channel(s) to measure resistance and adds the channel(s) to the task you specify with **taskHandle**.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you mus names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> ; the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you	expect to meas	ure.
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you	expect to meas	sure.
units	int32	The units to use to return the measur	ement.	
		Name		Description
		DAQmx_Val_Ohms		ohms
		DAQmx_Val_FromCustomScale Units sp a custom Use <b>custom</b> to specia scale.		Units specifie a custom scale Use <b>customScale</b> to specify a cu scale.
resistanceConfig	int32	The configuration for resistance mea	surements.	
		Value	Descript	ion
		DAQmx_Val_2Wire	2-wire m	ode.
		DAQmx_Val_3Wire	3-wire m	ode.
		DAQmx_Val_4Wire 4-wire mode.		ode.
currentExcitSource	int32	The source of excitation.		
		Value	Descri	ption
		DAQmx_Val_Internal	Use the source must u to spec	e built-in excita of the device. se <b>currentExc</b> ify the amount

excitation.

	DAQmx_Val_External	Use an excitation sour other than the built-in excitation source of th device. You must use <b>currentExcitVal</b> to specify the amount of excitation.		
	DAQmx_Val_None	Supply no excitation the channel. You cannot up this value if the senso requires excitation.		
float64	The amount of excitation, in amperes, that the sensor requires.			
const char []	The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set <b>units</b> to DAOmx. Val. FromCustomScale. If you do not set <b>unit</b>			

currentExcitVal customScaleName

must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **uni** DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to N

#### Name Type Description

# DAQmxCreateAIRTDChan

int32 DAQmxCreateAIRTDChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 rtdType, int32 resistanceConfig, int32 currentExcitSource, float64 currentExcitVal, float64 r0);

Creates channel(s) that use an <u>RTD</u> to measure temperature and adds the channel(s) to the task you specify with **taskHandle**.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channe You can specify a <u>list or range</u> of phy	ls to use to create virtual channels. ysical channels.	
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do no specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
minVal	float64	The <u>minimum value</u> , in <b>units</b> , that yo	u expect to measure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that yo	ou expect to measure.	
units	int32	The units to use to return the meas	surement.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	
rtdType	int32	The type of RTD connected to the ch	annel.	
		Value	Description	
		DAQmx_Val_Pt3750	Pt3750	
		DAQmx_Val_Pt3851	Pt3851	
		DAQmx_Val_Pt3911	Pt3911	
		DAQmx_Val_Pt3916	Pt3916	
		DAQmx_Val_Pt3920	Pt3920	
		DAQmx_Val_Pt3928	Pt3928	
		DAQmx_Val_Custom	You must specify the 'A', 'B', and 'C' constants for the <u>Callendar-</u> <u>Van Dusen equation</u>	

resistanceConfig	int32	The configuration for resistance measurements.		
		Value	Description	
		DAQmx_Val_2Wire	2-wire mode.	
		DAQmx_Val_3Wire	3-wire mode.	
		DAQmx_Val_4Wire	4-wire mode.	
currentExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excitation source of the device. You must use <b>currentExcitVal</b> to specify the amount of excitation.	
		DAQmx_Val_External	Use an excitation source other than the built-in excitation source of the device. You must use <b>currentExcitVal</b> to specify the amount of excitation.	
		DAQmx_Val_None	Supply no excitation to the channel. You cannot use this value if the sensor requires excitation.	
currentExcitVal	float64	The amount of excitation, in ampere	s, that the sensor requires.	
r0	float64	The sensor resistance in ohms at 0 deg C for the <u>Callendar-Van Dusen</u> $\frac{1}{2}$		

#### Name Type Description

# DAQmxCreateAIStrainGageChan

int32 DAQmxCreateAIStrainGageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 strainConfig, int32 voltageExcitSource, float64 voltageExcitVal, float64 gageFactor, float64 initialBridgeVoltage, float64 nominalGageResistance, float64 poissonRatio, float64 leadWireResistance, const char customScaleName[]);

Creates channel(s) to measure strain and adds the channel(s) to the task you specify with **taskHandle**.

Name	Туре	Description			
taskHandle	TaskHandle	The task to which to add the channels that this function creates.			
physicalChannel	const char []	The names of the physical channels to use to create specify a <u>list or range</u> of physical channels.	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you mus names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.			
minVal	float64	The minimum value, in <b>units</b> , that you expect to measure	sure.		
maxVal	float64	The maximum value, in <b>units</b> , that you expect to measure	sure.		
units	int32	The units to use to return the measurement.			
		Name	Description		
		DAQmx_Val_Strain	Strain		
		DAQmx_Val_FromCustomScale	Units specifie a custom scale Use <b>customScale!</b> to specify a cu scale.		
strainConfig	int32	The strain gage bridge configuration.			
		Value	Description		
		DAQmx_Val_FullBridgeI DAQmx_Val_FullBridgeII	Four active ga with two pairs subjected to ea and opposite strains. Four active ga		
			with two align with maximur principal strait two Poisson g		

				in adjacent arı
		DAQmx_Val_FullBridg	eIII	Four active ga with two align with maximur principal strai two Poisson g in opposite an
		DAQmx_Val_HalfBridg	jeI	Two active ga with one align with maximur principal strain one Poisson g
		DAQmx_Val_HalfBridg	jeII	Two active ga with equal and opposite strair
		DAQmx_Val_QuarterBr	ridgeI	Single active {
		DAQmx_Val_QuarterBr	ridgeII	Single active { and one dumn gage.
voltageExcitSource	int32	The source of excitation.		
		Value	Descr	ription
		DAQmx_Val_Internal	Use th source must to spe excita	ne built-in excita e of the device. ` use <b>voltageExci</b> cify the amount tion.
		DAQmx_Val_External	Use a other excita device <b>voltag</b> the an	n excitation sou than the built-in tion source of th e. You must use geExcitVal to sp nount of excitati
		DAQmx_Val_None	Suppl chann this va	y no excitation ( el. You cannot u alue if the senso

# requires excitation.

voltageExcitVal	float64	The amount of excitation, in volts, that the sensor requires.	
gageFactor	float64	The sensitivity of the strain gages and relates the change in electrical res to the change in strain. Each gage in the bridge must have the same gage Refer to the sensor documentation to determine this value.	
initialBridgeVoltage	float64	The bridge output voltage in the unloaded condition. NI-DAQmx subtra value from any measurements before applying scaling equations. Perfor voltage measurement on the bridge with no strain applied to determine t value.	
nominalGageResistance	float64	The resistance, in ohms, of the gages in an unstrained position. Each gag bridge must have the same nominal gage resistance. The resistance acro of the bridge that do not have strain gages must also be the same as the 1 gage resistance. Refer to the sensor documentation to determine this val	
poissonRatio	float64	The ratio of lateral strain to axial strain in the material in which you meastrain.	
leadWireResistance	float64	The amount, in ohms, of resistance in the lead wires. Ideally, this value i same for all leads.	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set <b>units</b> to DAQmx_Val_FromCustomScale. If you do not set <b>uni</b> DAQmx_Val_FromCustomScale, you must set <b>customScaleName</b> to N	

#### Name Type Description

# DAQmxCreateAITempBuiltInSensorChan

Creates channel(s) to measure temperature with a built-in sensor of a terminal block or device and adds the channel(s) to the task you specify with **taskHandle**. On SCXI modules, for example, this could be the CJC sensor.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI- DAQmx <u>automatically assigns names</u> to the virtual channels.		
units	int32	The units to use to return the measurem	nent.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	

#### Name Type Description

# DAQmxCreateAIThrmcplChan

int32 DAQmxCreateAIThrmcplChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 thermocoupleType, int32 cjcSource, float64 cjcVal, const char cjcChannel[]);

Creates channel(s) that use a *thermocouple* to measure temperature and adds the channel(s) to the task you specify with **taskHandle**.

Name	Туре	Description			
taskHandle	TaskHandle	The task to which to add the channels that this function creates.			
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. You can specify a $\underline{list \ or \ range}$ of physical channels.			
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.			
minVal	float64	The <i>minimum value</i> , in <b>units</b> , that you expect to measure.			
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to measure.			
units	int32	The units to use to return the measurement.			
		Name	Desc	cription	
		DAQmx_Val_DegC	degr	ees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit		
		DAQmx_Val_Kelvins	DAQmx_Val_Kelvins kelvins DAQmx_Val_DegR degrees Rankine		
		DAQmx_Val_DegR			
thermocoupleType	int32	The $\underline{type}$ of thermocouple connected to	couple connected to the channel.		
		Value		Description	
		DAQmx_Val_J_Type_TC		J-type thermocouple.	
		DAQmx_Val_K_Type_TC		K-type thermocouple.	
		DAQmx_Val_N_Type_TC		N-type thermocouple.	
		DAQmx_Val_R_Type_TC		R-type thermocouple.	
		DAQmx_Val_S_Type_TC	1	S-type thermocouple.	

		DAQmx_Val_T_Type_TC	T-type thermocouple.	
		DAQmx_Val_B_Type_TC	B-type thermocouple.	
		DAQmx_Val_E_Type_TC	E-type thermocouple.	
cjcSource	int32	The source of <u>cold junction compensation</u> .		
		Value	Description	
		DAQmx_Val_BuiltIn	Use a cold- junction compensation channel built into the terminal block.	
		DAQmx_Val_ConstVal	You must specify the cold-junction temperature.	
		DAQmx_Val_Chan	Use a channel for cold-junction compensation.	
cjcVal	float64	The temperature of the cold junction of the thermocouple if you set <b>cjcSource</b> to DAQmx_Val_ConstVal.		
cjcChannel	const char []	The channel that acquires the temperature of the thermocouple <u>cold-junction</u> if you set <b>cjcSource</b> to DAQmx_Val_Chan. You can use a global channel or another virtual channel already in the task. 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.		

#### Name Type Description

# DAQmxCreateAIThrmstrChanIex

int32 DAQmxCreateAIThrmstrChanIex (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 currentExcitSource, float64 currentExcitVal, float64 a, float64 b, float64 c);

Creates channel(s) that use a *thermistor* to measure temperature and adds the channel(s) to the task you specify with **taskHandle**. Use this function when the thermistor requires current excitation.
Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels You can specify a <u>list or range</u> of physi	to use to create virtual channels. cal channels.	
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do no specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx automatically assigns names to the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you	expect to measure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you	expect to measure.	
units	int32	The units to use to return the measur	ement.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	
resistanceConfig	int32	The configuration for resistance mea	surements.	
		Value	Description	
		DAQmx_Val_2Wire	2-wire mode.	
		DAQmx_Val_3Wire	3-wire mode.	
		DAQmx_Val_4Wire	4-wire mode.	
currentExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excitation source of the device. You must use <b>currentExcitVal</b>	

		to specify the amount of excitation.
	DAQmx_Val_External	Use an excitation source other than the built-in excitation source of the device. You must use <b>currentExcitVal</b> to specify the amount of excitation.
	DAQmx_Val_None	Supply no excitation to the channel. You cannot use this value if the sensor requires excitation.
nt64	The amount of excitation, in amperes	, that the sensor requires.
1t64	The A constant from the Steinhart-Hart the Steinhar	hermistor equation.

currentExcitVal	float64	The amount of excitation, in amperes, that the sensor requires
a	float64	The A constant from the Steinhart-Hart thermistor equation.
b	float64	The B constant from the Steinhart-Hart thermistor equation.
C	float64	The C constant from the <u>Steinhart-Hart thermistor equation</u> .

#### Name Type Description

# DAQmxCreateAIThrmstrChanVex

int32 DAQmxCreateAIThrmstrChanVex (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 voltageExcitSource, float64 voltageExcitVal, float64 a, float64 b, float64 c, float64 r1);

## Purpose

Creates channel(s) that use a *thermistor* to measure temperature and adds the channel(s) to the task you specify with **taskHandle**. Use this function when the thermistor requires voltage excitation.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to You can specify a <u>list or range</u> of physical channels to the physical channels of	to use to create virtual channels. cal channels.	
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do no specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you e	expect to measure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you	expect to measure.	
units	int32	The units to use to return the measur	ement.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	
resistanceConfig	int32	The configuration for resistance mea	surements.	
		Value	Description	
		DAQmx_Val_2Wire	2-wire mode.	
		DAQmx_Val_3Wire	3-wire mode.	
		DAQmx_Val_4Wire	4-wire mode.	
voltageExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excitation source of the device. You must use <b>voltageExcitVal</b> to	

	specify the amount of excitation.
DAQmx_Val_External	Use an excitation source other than the built-in excitation source of the device. You must use <b>voltageExcitVal</b> to
	specify the amount of excitation.
DAQmx_Val_None	Supply no excitation to the channel. You cannot use this value if the sensor

requires excitation.

voltageExcitVal	float64	The amount of excitation, in volts, that the sensor requires.
a	float64	The A constant from the Steinhart-Hart thermistor equation.
b	float64	The B constant from the Steinhart-Hart thermistor equation.
C	float64	The C constant from the <u>Steinhart-Hart thermistor equation</u> .
r1	float64	The value, in ohms, of the reference resistor.

#### Name Type Description

# DAQmxCreateAIVoltageChan

int32 DAQmxCreateAIVoltageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, const char customScaleName[]);

## Purpose

Creates channel(s) to measure voltage and adds the channel(s) to the task you specify with **taskHandle**. If your measurement requires the use of internal excitation or you need the voltage to be scaled by excitation, call <u>DAQmxCreateAIVoltageChanWithExcit</u>.

Input

Name	Туре	Description	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.			
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.			
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.			
terminalConfig	int32	The <i>input terminal configuration</i> for the channel.			
		Value I	Description		
		DAQmx_Val_Cfg_Default A (-1)	At run time, NI- DAQmx chooses efault terminal configuration f		
		ť	he channel.		
		DAQmx_Val_RSE	eferenced single-ended mode		
		DAQmx_Val_NRSE	onreferenced single-ended mo		
		DAQmx_Val_Diff	ifferential mode		
		DAQmx_Val_PseudoDiff	seudodifferential mode		
minVal	float64	The minimum value, in <b>units</b> , that you expect to me	easure.		
maxVal	float64	The maximum value, in <b>units</b> , that you expect to m	easure.		
units	int32	The units to use to return the voltage measureme	ents.		
		Name	Description		
		DAQmx_Val_Volts	volts		
		DAQmx_Val_FromCustomScale	e Units a custon scale specifies <b>customScale!</b> to specify a cu scale.		

customScaleName

const char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **un**i DAQmx\_Val\_FromCustomScale, you must set customScaleName to N

#### Name Type Description

# DAQmxCreateAIVoltageRMSChan

int32 DAQmxCreateAIVoltageRMSChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, const char customScaleName[]);

## Purpose

Creates channel(s) to measure RMS voltage and adds the channel(s) to the task you specify with **taskHandle**. If your measurement requires the use of internal excitation or you need the voltage to be scaled by excitation, call DAQmxCreateAIVoltageChanWithExcit.

Input

Name	Туре	Description	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.			
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.			
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.			
terminalConfig	int32	The <i>input terminal configuration</i> for the channel.			
		Value I	Description		
		DAQmx_Val_Cfg_Default A (-1)	At run time, NI- DAQmx chooses efault terminal configuration f		
		ť	he channel.		
		DAQmx_Val_RSE	eferenced single-ended mode		
		DAQmx_Val_NRSE	onreferenced single-ended mo		
		DAQmx_Val_Diff	ifferential mode		
		DAQmx_Val_PseudoDiff	seudodifferential mode		
minVal	float64	The minimum value, in <b>units</b> , that you expect to me	easure.		
maxVal	float64	The maximum value, in <b>units</b> , that you expect to m	easure.		
units	int32	The units to use to return the voltage measureme	ents.		
		Name	Description		
		DAQmx_Val_Volts	volts		
		DAQmx_Val_FromCustomScale	e Units a custon scale specifies <b>customScale!</b> to specify a cu scale.		

customScaleName

const char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **un**i DAQmx\_Val\_FromCustomScale, you must set customScaleName to N

#### Name Type Description

# DAQmxCreateAIVoltageChanWithExcit

int32 DAQmxCreateAIVoltageChanWithExcit (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, int32 bridgeConfig, int32 voltageExcitSource, float64 voltageExcitVal, bool32 useExcitForScaling, const char customScaleName[]);

## Purpose

Creates channel(s) to measure voltage and adds the channels to the task you specify with **taskHandle**. Use this instance for custom sensors that require excitation. You can choose to use the excitation to scale the measurement.

Input

Name	Туре	Description	Description		
taskHandle	TaskHandle	The task to which to add the channels that this	func	tion creates.	
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.			
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.			
terminalConfig	int32	The input terminal configuration for the channel.			
		Value	De	escription	
		DAQmx_Val_Cfg_Default (-1) DAQmx_Val_RSE DAQmx_Val_NRSE DAQmx_Val_Diff DAQmx_Val_PseudoDiff	At DA defaul the Reference Nonree Differ	run time, NI- Qmx chooses (t terminal configuration f channel. enced single-ended mode efferenced single-ended mode rential mode	
minVal	float64	The minimum value, in <b>units</b> , that you expect to r	meası	ure.	
maxVal	float64	The maximum value, in <b>units</b> , that you expect to a	meas	ure.	
units	int32	The units to use to return the voltage measurer	ments	5.	
		Name		Description	
		DAQmx_Val_Volts		volts	
		DAQmx_Val_FromCustomSca	le	Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
bridgeConfig	int32	The type of Wheatstone bridge the sensor is.			

#### Value

Description DAQmx\_Val\_FullBridge Sensor is a full bridge. If you se useExcitForSci to TRUE, NI-DAQmx divide: measurement by excitation value Many sensors so data to native u using scaling of volts per excitat DAQmx\_Val\_HalfBridge Sensor is a half bridge. If you se useExcitForSci to TRUE, NI-DAQmx divide: measurement by excitation value Many sensors so data to native u using scaling of volts per excitat Sensor is a quar bridge. If you se useExcitForSci to TRUE, NI-DAQmx divide: measurement by excitation value Many sensors so data to native u using scaling of volts per excitat Sensor is not a Wheatstone brid

DAQmx\_Val\_QuarterBridge

#### DAQmx\_Val\_NoBridge

The source of excitation.

		Value	Description
		DAQmx_Val_Internal	Use the built-in excita source of the device. ' must use <b>voltageExci</b> to specify the amount excitation.
		DAQmx_Val_External	Use an excitation sou other than the built-in excitation source of th device. You must use <b>voltageExcitVal</b> to sp the amount of excitati
		DAQmx_Val_None	Supply no excitation t channel. You cannot t this value if the senso requires excitation.
voltageExcitVal	float64	The amount of excitation, in volts, that t	the sensor requires.
useExcitForScaling	bool32	<ul> <li>Specifies whether NI-DAQmx divides the measurement by the exc should typically set useExcitForScaling to TRUE for ratiometric to you set useExcitForScaling to TRUE, set maxVal and minVal to scaling.</li> <li>For example, if you expect to acquire a voltage between -5 and 5, a excitation of .10 volts to scale the measurement, set minVal to -50 maxVal to 50. If you set bridgeConfig to DAQmx_Val_NoBridge useExcitForScaling has no effect on the measurement.</li> </ul>	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the must set <b>units</b> to DAQmx_Val_FromCustomScale, you n	e channel. To use this parameter, istomScale. If you do not set <b>uni</b> nust set <b>customScaleName</b> to N

#### Name Type Description

# DAQmxCreateAIPosLVDTChan

int32 DAQmxCreateAIPosLVDTChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, float64 sensitivity, int32 sensitivityUnits, int32 voltageExcitSource, float64 voltageExcitVal, float64 voltageExcitFreq, int32 ACExcitWireMode, const char customScaleName[]);

# Purpose

Creates channel(s) that use an <u>LVDT</u> to measure linear position and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description			
taskHandle	TaskHandle	The task to which to add the channels that this function creates.			
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo range of physical channels.			els. Yo
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spe DAQmx uses the physical channel name as the virtual channel name. If names for <b>nameToAssignToChannel</b> , you must use the names when yc channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you names separated by commas. If you provide fewer names than the numb you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.			
minVal	float64	The minimum value, in <b>units</b> , that you expe	ect to measure.		
maxVal	float64	The maximum value, in <b>units</b> , that you exp	ect to measure.		
units	int32	The units to use to return linear position	measurements	from the	channe
		Name		Descri	ptioı
		DAQmx_Val_Meters		Meters	
		DAQmx_Val_Inches I		Inches	
		DAQmx_Val_FromCustomScale Units a specifie <b>custom</b> specify		es. U n <b>Sca</b> 7 a cu	
sensitivity	float64	The sensitivity of the sensor. This value to the sensor documentation to determine	is in the units get this value.	you specif	y with
sensitivityUnits	int32	The units of <b>sensitivity</b> .			
		Name			Des
		DAQmx_Val_mVoltsPerVo	oltPerMilli	meter	mvc
		DAQmx_Val_mVoltsPerVo	oltPerMilli	Inch	mvc inch
voltageExcitSource	int32	The source of excitation.			
		Value	Descripti	on	
		DAQmx_Val_Internal	Use the bitthe device voltageEx amount of	uilt-in e e. You r <b>xcitVal</b> f excita	excita nust to sp tion.

		DAQmx_Val_External	Use an excitation sour the built-in excitation device. You must use <b>voltageExcitVal</b> to sp amount of excitation.
		DAQmx_Val_None	Supply no excitation t You cannot use this va sensor requires excita
voltageExcitVal	float64	The amount of excitation, in volts, th	at the sensor requires.
voltageExcitFreq	float64	The excitation frequency, in hertz, the to determine this value.	at the sensor requires. Refer to the
ACExcitWireMode	int32	The number of leads on the sensor. S a 4-wire or 5-wire sensor. Refer to the	ome sensors may require you to tie e documentation for your sensor fo
		Value	Description
		DAQmx_Val_4Wire	4-wire.
		DAQmx_Val_5Wire	5-wire.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to DAQmx_Val_FromCustomScale. If you must set <b>customScaleName</b> to N	the channel. To use this parameter, you do not set <b>units</b> to DAQmx_V& NULL.

#### Name Type Description

# DAQmxCreateAIPosRVDTChan

int32 DAQmxCreateAIPosRVDTChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, float64 sensitivity, int32 sensitivityUnits, int32 voltageExcitSource, float64 voltageExcitVal, float64 voltageExcitFreq, int32 ACExcitWireMode, const char customScaleName[]);

# Purpose

Creates channel(s) that use an <u>RVDT</u> to measure angular position and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spe DAQmx uses the physical channel name as the virtual channel name. If own names for <b>nameToAssignToChannel</b> , you must use the names whe these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you list of names separated by commas. If you provide fewer names than the virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the vi		
minVal	float64	The minimum value, in <b>units</b> , that you expect to measure.		
maxVal	float64	The maximum value, in <b>units</b> , that you expect to measure.		
units	int32	The units to use to return angular position measurements from the chant		
		Name DAQmx_Val_Degrees DAQmx_Val_Radians		Descriptio
				Degrees
				Radians
		DAQmx_Val_FromCustom	iScale	Units specific custom scal customSca specify a cuscale.
sensitivity	float64	The sensitivity of the sensor. This value is in the units you specify with Refer to the sensor documentation to determine this value.		
sensitivityUnits	int32	The units of <b>sensitivity</b> .		
		Name		Descriț
		DAQmx_Val_mVoltsPerVoltPerDegree mvolts/		
		DAQmx_Val_mVoltsPerVo	mx_Val_mVoltsPerVoltPerRadian mvolt	
voltageExcitSource	int32	The source of excitation.		
		Value	Descripti	on
DAQmx_Va		DAQmx_Val_Internal	Use the built-in excitation source of the device. The source of the device of the source of the amount of excitation of excitati	

		DAQmx_Val_External	Use an excitation sou than the built-in excit source of the device. use <b>voltageExcitVal</b> t the amount of excitati	
		DAQmx_Val_None	Supply no excitation the channel. You cannot under the sensor recent the sensor rece	
voltageExcitVal	float64	The amount of excitation, in volts, that the sensor requires.		
voltageExcitFreq	float64	The excitation frequency, in hertz, that the sensor requires. Refer to the a documentation to determine this value.		
ACExcitWireMode	int32	The number of leads on the sensor. Some sensors may require you to the create a 4-wire or 5-wire sensor. Refer to the documentation for your se information.		
		Value	Description	
		DAQmx_Val_4Wire	4-wire.	
		DAQmx_Val_5Wire	5-wire.	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to <b>units</b> to DAQmx_Val_FromCustomS DAQmx_Val_FromCustomScale, yo	the channel. To use this parameter, Scale. If you do not set <b>units</b> to u must set <b>customScaleName</b> to N	

#### Name Type Description

# Obsolete

This function is obsolete. Use <u>DAQmxCreateAITempBuiltInSensorChan</u> instead.

# DAQmxCreateAIDeviceTempChan

int32 DAQmxCreateAIDeviceTempChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 units);
Creates channel(s) that use a sensor built into a terminal block or device to measure temperature and adds the channel(s) to the task you specify with **taskHandle**. On SCXI modules, for example, this could be the CJC sensor.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
units	int32	The units to use to return the measurem	nent.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	

#### Name Type Description

# DAQmxCreateTEDSAIAccelChan

int32 DAQmxCreateTEDSAIAccelChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, int32 currentExcitSource, float64 currentExcitVal, const char customScaleName[]);

Creates channel(s) that use an <u>accelerometer</u> to <u>measure acceleration</u> and adds the channel(s) to the <u>task</u> you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
terminalConfig	int32	The input terminal configuration for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At DA	run time, NI- Qmx chooses
			defau the	<u>It terminal configuration</u> f channel.
		DAQmx_Val_RSE	<u>Refer</u>	enced single-ended mode
		DAQmx_Val_NRSE	Nonre	eferenced single-ended mo
		DAQmx_Val_Diff	Diffe	<u>rential mode</u>
		DAQmx_Val_PseudoDiff	<u>Pseud</u>	lodifferential mode
minVal	float64	The minimum value, in <b>units</b> , that you expect to	meas	ure.
maxVal	float64	The $\underline{\text{maximum value}}$ , in <b>units</b> , that you expect to	meas	ure.
units	int32	The units to use to return acceleration measur	emen	ts from the channel.
		Name		Description
		DAQmx_Val_AccelUnit_g		G. 1 g is approximately equal to 9.81 1
		DAQmx_Val_FromCustomSca	le	Units specifie a custom scale Use <b>customScale</b> to specify a cu

scale.

currentExcitSource	int32	The source of excitation.	
		Value	Description
		DAQmx_Val_Internal	Use the built-in excitation source of the device. The must use <b>currentExc</b> to specify the amount excitation.
		DAQmx_Val_External	Use an excitation sour other than the built-in excitation source of th device. You must use <b>currentExcitVal</b> to specify the amount of excitation.
		DAQmx_Val_None	Supply no excitation the channel. You cannot up this value if the senso requires excitation.
currentExcitVal	float64	The amount of excitation, in amperes,	that the sensor requires.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the must set <b>units</b> to DAQmx_Val_FromCDAQmx_Val_FromCustomScale, you	ne channel. To use this parameter, SustomScale. If you do not set <b>un</b> i must set <b>customScaleName</b> to N

#### Name Type Description

# DAQmxCreateTEDSAICurrentChan

int32 DAQmxCreateTEDSAICurrentChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, int32 shuntResistorLoc, float64 externalShuntResistorValue, const char customScaleName[]);

Creates channel(s) to measure current and adds the channel(s) to the task you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a $list or range$ of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channel If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
terminalConfig	int32	The input terminal configuration for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At DA defau the	run time, NI- Qmx chooses It terminal configuration f channel.
		DAOmx Val RSE	<u>Refer</u>	enced single-ended mode
		DAOmy Val NRSE	Nonr	eferenced single-ended mo
		DAOmy Val Diff	Diffe	rential mode
			Pseuc	lodifferential mode
		DAQmx_Val_PseudoDiff	<u>r seue</u>	
minVal	float64	The $\underline{\mbox{minimum value}},$ in <b>units</b> , that you expect to	meas	ure.
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to	meas	sure.
units	int32	The units to use to return measurements.		
		Name		Description
		DAQmx_Val_FromCustomSca	ale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.
		DAQmx_Val_FromTEDS		Units defined TEDS informa associated wit

channel.

shuntResistorLoc	int32	The location of the shunt resistor.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in shunt resistor of the device.	
		DAQmx_Val_External	Use a shunt resistor external to the device must specify the value the shunt resistor in <b>extShuntResistorVal</b>	
extShuntResistorVal	float64	The value, in ohms, of an external shund	t resistor.	
customScaleName	const char []	[] The name of a <u>custom scale</u> to apply to the channel. To use this parameter must set <b>units</b> to DAQmx_Val_FromCustomScale. If you do not set <b>un</b> DAQmx_Val_FromCustomScale, you must set <b>customScaleName</b> to I		

#### Name Type Description

# DAQmxCreateTEDSAIMicrophoneChan

int32 DAQmxCreateTEDSAIMicrophoneChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, int32 units, float64 maxSndPressLevel, int32 currentExcitSource, float64 currentExcitVal, const char customScaleName[]);

Creates channel(s) that use a microphone to measure sound pressure and adds the channel(s) to the task you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
terminalConfig	int32	The input terminal configuration for the channel.		
		Value	De	scription
		DAQmx_Val_Cfg_Default (-1)	At DA	run time, NI- Qmx chooses
			<sub>defau</sub>	<u>lt terminal configuration</u> f channel.
		DAQmx_Val_RSE	<u>Refer</u>	enced single-ended mode
		DAQmx_Val_NRSE	Nonre	eferenced single-ended mo
		DAQmx_Val_Diff	<u>Diffe</u>	rential mode
		DAQmx_Val_PseudoDiff	Pseud	lodifferential mode
units	int32	The units to use to return sound pressure mea	surem	ients.
		Name		Description
		DAQmx_Val_Pascals		pascals
		DAQmx_Val_FromCustomSca	ale	Units specifie a custom scale Use <b>customScale</b> to specify a cu scale.
maxSndPressLevel	float64	The maximum instantaneous sound pressure value is in decibels, referenced to 20 micropa	level y scals.	you expect to measur
currentExcitSource	int32	The source of excitation.		

		Value	Description
		DAQmx_Val_Internal	Use the built-in excita source of the device. must use <b>currentExc</b> to specify the amount excitation.
		DAQmx_Val_External	Use an excitation sour other than the built-in excitation source of th device. You must use <b>currentExcitVal</b> to specify the amount of excitation.
		DAQmx_Val_None	Supply no excitation t channel. You cannot t this value if the senso requires excitation.
currentExcitVal	float64	The amount of excitation, in amperes, th	nat the sensor requires.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the must set <b>units</b> to DAQmx_Val_FromCu DAQmx_Val_FromCustomScale, you n	channel. To use this parameter, stomScale. If you do not set <b>un</b> i nust set <b>customScaleName</b> to N

#### Name Type Description

# DAQmxCreateTEDSAIResistanceChan

int32 DAQmxCreateTEDSAIResistanceChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 currentExcitSource, float64 currentExcitVal, const char customScaleName[]);

Creates channel(s) to measure resistance and adds the channel(s) to the task you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The <i>minimum value</i> , in <b>units</b> , that you	expect to meas	ure.
maxVal	float64	The maximum value, in <b>units</b> , that you	expect to meas	sure.
units	int32	The units to use to return measureme	ents.	
		Name		Description
		DAQmx_Val_FromCustomScale Uni scal cus to s scal DAQmx_Val_FromTEDS Uni TEJ asso cha		Units a custon scale specifies <b>customScale</b> ? to specify a cu scale. Units defined
				TEDS information associated wit channel.
resistanceConfig	int32	The configuration for resistance mea	asurements.	
		Value	Descript	ion
		DAQmx_Val_2Wire	2-wire m	ode.
		DAQmx_Val_3Wire	3-wire m	ode.
		DAQmx_Val_4Wire	4-wire m	ode.
currentExcitSource	int32	The source of excitation.		
		Value	Descri	ption
		DAQmx_Val_Internal	Use the source	e built-in excita of the device.

			must use <b>currentExc</b> to specify the amount excitation.
		DAQmx_Val_External	Use an excitation sour other than the built-in excitation source of the device. You must use <b>currentExcitVal</b> to specify the amount of excitation.
		DAQmx_Val_None	Supply no excitation t channel. You cannot t this value if the senso requires excitation.
currentExcitVal	float64	The amount of excitation, in amperes, th	nat the sensor requires.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the must set <b>units</b> to DAQmx_Val_FromCuDAQmx_Val_FromCustomScale, you n	e channel. To use this parameter, istomScale. If you do not set <b>un</b> i nust set <b>customScaleName</b> to N

#### Name Type Description

# DAQmxCreateTEDSAIRTDChan

int32 DAQmxCreateTEDSAIRTDChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 currentExcitSource, float64 currentExcitVal);

Creates channel(s) that use an <u>RTD</u> to <u>measure temperature</u> and adds the channel(s) to the <u>task</u> you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

-				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels You can specify a <u>list or range</u> of physi	to use to create virtual channels. cal channels.	
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refit to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx automatically assigns names to the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you	expect to measure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you	expect to measure.	
units	int32	The units to use to return the measur	rement.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	
resistanceConfig	int32	The configuration for resistance mea	asurements.	
		Value	Description	
		DAQmx_Val_2Wire	2-wire mode.	
		DAQmx_Val_3Wire	3-wire mode.	
		DAQmx_Val_4Wire	4-wire mode.	
currentExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excitation source of the device. You must use <b>currentExcitVal</b>	

			to specify the amount of excitation.
		DAQmx_Val_External	Use an excitation source other than the built-in excitation source of the device. You must use <b>currentExcitVal</b> to specify the amount of excitation.
		DAQmx_Val_None	Supply no excitation to the channel. You cannot use this value if the sensor requires excitation.
currentExcitVal	float64	The amount of excitation, in amperes,	that the sensor requires.

#### Name Type Description

# DAQmxCreateTEDSAIStrainGageChan

int32 DAQmxCreateTEDSAIStrainGageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 voltageExcitSource, float64 voltageExcitVal, float64 initialBridgeVoltage, float64 leadWireResistance, const char customScaleName[]);

Creates channel(s) to measure strain and adds the channel(s) to the task you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you expect to measure.		
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to measure.		
units	int32	The units to use to return the measurement.		
		Name		Description
		DAQmx_Val_Strain		Strain
		DAQmx_Val_FromCustom	ıScale	Units specifie a custom scale Use <b>customScale</b> to specify a cu scale.
voltageExcitSource	int32	The source of excitation.		
		Value	<b>Description</b> Use the built-in excita source of the device. The must use <b>voltageExci</b> to specify the amount excitation.	
		DAQmx_Val_Internal		
		DAQmx_Val_External	Use an other th excitati device. <b>voltage</b>	excitation sou an the built-in on source of th You must use ExcitVal to sp

### DAQmx\_Val\_None

the amount of excitati

Supply no excitation t channel. You cannot u this value if the senso requires excitation.

voltageExcitVal	float64	The amount of excitation, in volts, that the sensor requires.
initialBridgeVoltage	float64	The bridge output voltage in the unloaded condition. NI-DAQmx subtra value from any measurements before applying scaling equations. Perfor voltage measurement on the bridge with no strain applied to determine t value.
leadWireResistance	float64	The amount, in ohms, of resistance in the lead wires. Ideally, this value same for all leads.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set <b>units</b> to DAQmx_Val_FromCustomScale. If you do not set <b>uni</b> DAQmx_Val_FromCustomScale, you must set <b>customScaleName</b> to N

#### Name Type Description

# DAQmxCreateTEDSAIThrmcplChan

int32 DAQmxCreateTEDSAIThrmcplChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 cjcSource, float64 cjcVal, const char cjcChannel[]);

Creates channel(s) that use a thermocouple to measure temperature and adds the channel(s) to the task you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Name	Туре	Description			
taskHandle	TaskHandle	The task used in this function.			
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. You can specify a $list or range$ of physical channels.			
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx automatically assigns names to the virtual channels.			
minVal	float64	The minimum value, in <b>units</b> , that you expect to measure.			
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to measure.			
units	int32	The units to use to return the measurement.			
		Name	Description		
		DAQmx_Val_DegC	degrees Celsius		
		DAQmx_Val_DegF	degrees Fahrenheit		
		DAQmx_Val_Kelvins	kelvins		
		DAQmx_Val_DegR	degrees Rankine		
cjcSource	int32	The source of <u>cold junction compensation</u> .	cold junction compensation.		
		Value	Description		
		DAQmx_Val_BuiltIn	Use a cold- junction compensation channel built into the terminal block.		
		DAQmx_Val_ConstVal	You must specify the cold-junction temperature.		
		DAQmx_Val_Chan	Use a channel for cold-junction		
# compensation.

cjcVal	float64	The temperature of the cold junction of the thermocouple if you set <b>cjcSource</b> to DAQmx_Val_ConstVal.
cjcChannel	const char []	The channel that acquires the temperature of the thermocouple <u>coldjunction</u> if you set <b>cjcSource</b> to DAQmx_Val_Chan. You can use a global channel or another virtual channel already in the task. 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.

#### Name Type Description

# DAQmxCreateTEDSAIThrmstrChanIex

int32 DAQmxCreateTEDSAIThrmstrChanIex (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 currentExcitSource, float64 currentExcitVal);

Creates channel(s) that use a thermistor to measure temperature and adds the channel(s) to the task you specify with **taskHandle**. Use this instance when the thermistor requires current excitation. You must configure the physical channel(s) with TEDS information to use this function.

Input

-				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do n specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refet to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function you can specify a list of names separated by commas. If you provid fewer names than the number of virtual channels you create, NI-DAOmx automatically assigns names to the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you	expect to measure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you	expect to measure.	
units	int32	The units to use to return the measur	rement.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	
resistanceConfig	int32	The configuration for resistance mea	asurements.	
		Value	Description	
		DAQmx_Val_2Wire	2-wire mode.	
		DAQmx_Val_3Wire	3-wire mode.	
		DAQmx_Val_4Wire	4-wire mode.	
currentExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excitation source of the device. You must use <b>currentExcitVal</b>	

			to specify the amount of excitation.
		DAQmx_Val_External	Use an excitation source other than the built-in excitation source of the device. You must use <b>currentExcitVal</b> to specify the amount of excitation.
		DAQmx_Val_None	Supply no excitation to the channel. You cannot use this value if the sensor requires excitation.
currentExcitVal	float64	The amount of excitation, in amperes,	that the sensor requires.

#### Name Type Description

# DAQmxCreateTEDSAIThrmstrChanVex

int32 DAQmxCreateTEDSAIThrmstrChanVex (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 resistanceConfig, int32 voltageExcitSource, float64 voltageExcitVal, float64 r1);

Creates channel(s) that use a thermistor to measure temperature and adds the channel(s) to the task you specify with **taskHandle**. Use this instance when the thermistor requires voltage excitation. You must configure the physical channel(s) with TEDS information to use this function.

Input

-				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channel You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do n specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you ref to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function you can specify a list of names separated by commas. If you provi- fewer names than the number of virtual channels you create, NI- DAQmx automatically assigns names to the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you e	expect to measure.	
maxVal	float64	The maximum value, in <b>units</b> , that you	expect to measure.	
units	int32	The units to use to return the measur	ement.	
		Name	Description	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_DegR	degrees Rankine	
resistanceConfig	int32	The configuration for resistance mea	surements.	
		Value	Description	
		DAQmx_Val_2Wire	2-wire mode.	
		DAQmx_Val_3Wire	3-wire mode.	
		DAQmx_Val_4Wire	4-wire mode.	
voltageExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excitation source of the device. You must use <b>voltageExcitVal</b> to	

			specify the amount of excitation.
		DAQmx_Val_External	Use an excitation source other than the built-in excitation source of the device. You must use <b>voltageExcitVal</b> to specify the amount of excitation.
		DAQmx_Val_None	Supply no excitation to the channel. You cannot use this value if the sensor requires excitation.
voltageExcitVal	float64	The amount of excitation, in volts, the	at the sensor requires.
r1	float64	The value, in ohms, of the reference r	esistor.

#### Name Type Description

# DAQmxCreateTEDSAIVoltageChan

int32 DAQmxCreateTEDSAIVoltageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, const char customScaleName[]);

Creates channel(s) to measure voltage and adds the channel(s) to the task you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function. If your measurement requires the use of internal excitation or you need the voltage to be scaled by the excitation, use the

 $\underline{\text{DAQmxCreateTEDSAIVoltageChanWithExcit}} \ function.$ 

Input

Name	Туре	Description			
taskHandle	TaskHandle	The task used in this function.			
physicalChannel	const char []	The names of the physical channels to use to specify a <u>list or range</u> of physical channels.	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> the virtual channels.			
terminalConfig	int32	The input terminal configuration for the channel.			
		Value	De	scription	
		DAQmx_Val_Cfg_Default (-1)	At DA defau the	run time, NI- Qmx chooses It terminal configuration f channel.	
		$D \land Omv$ Val $RSF$		enced single-ended mode	
		DAOmy Val NRSE	Nonr	eferenced single-ended mo	
		DAQmx_Val_Diff DAQmx_Val_PseudoDiff Pseudodifferer		rential mode	
				lodifferential mode	
minVal	float64	The $\underline{\mbox{minimum value}},$ in <b>units</b> , that you expect to	meas	ure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to	meas	sure.	
units	int32	The units to use to return measurements.			
		Name		Description	
		DAQmx_Val_FromCustomSca	ale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
		DAQmx_Val_FromTEDS		Units defined TEDS informa associated wit	

### channel.

customScaleName

const char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **uni** DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to N

#### Name Type Description

# DAQmxCreateTEDSAIVoltageChanWithExcit

int32 DAQmxCreateTEDSAIVoltageChanWithExcit (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, int32 voltageExcitSource, float64 voltageExcitVal, const char customScaleName[]);

Creates channel(s) to measure voltage and adds the channel(s) to the task you specify with **taskHandle**. Use this instance for custom sensors that require excitation. You can use the excitation to scale the measurement. You must configure the physical channel(s) with TEDS information to use this function.

Input

Name	Туре	Description			
taskHandle	TaskHandle	The task used in this function.			
physicalChannel	const char []	The names of the physical channels to use to especify a <u>list or range</u> of physical channels.	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> the virtual channels.			
terminalConfig	int32	The input terminal configuration for the channel.			
		Value	De	scription	
		DAQmx_Val_Cfg_Default (-1)	At DA defau the	run time, NI- Qmx chooses It terminal configuration f channel.	
		$D \land Omv$ Val $RSF$		enced single-ended mode	
		DAOmy Val NRSE	Nonr	eferenced single-ended mo	
		DAQmx_Val_Diff DAQmx_Val_PseudoDiff Pseudodifferer		rential mode	
				lodifferential mode	
minVal	float64	The $\underline{\mbox{minimum value}},$ in <b>units</b> , that you expect to	meas	ure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to	meas	sure.	
units	int32	The units to use to return measurements.			
		Name		Description	
		DAQmx_Val_FromCustomSca	ale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
		DAQmx_Val_FromTEDS		Units defined TEDS informa associated wit	

channel.

voltageExcitSource	int32	The source of excitation.		
		Value	Description	
		DAQmx_Val_Internal	Use the built-in excita source of the device. must use <b>voltageExci</b> to specify the amount excitation.	
		DAQmx_Val_External	Use an excitation sour other than the built-in excitation source of the device. You must use <b>voltageExcitVal</b> to sp the amount of excitati	
		DAQmx_Val_None	Supply no excitation the channel. You cannot up this value if the senso requires excitation.	
voltageExcitVal	float64	The amount of excitation, in volts, that	the sensor requires.	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the must set <b>units</b> to DAQmx_Val_FromCustomScale, you n	e channel. To use this parameter, istomScale. If you do not set <b>un</b> i nust set <b>customScaleName</b> to N	

#### Name Type Description

# DAQmxCreateTEDSAIPosLVDTChan

int32 DAQmxCreateTEDSAIPosLVDTChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 voltageExcitSource, float64 voltageExcitVal, float64 voltageExcitFreq, int32 ACExcitWireMode, const char customScaleName[]);

Creates channel(s) that use an <u>LVDT</u> to <u>measure linear position</u> and adds the channel(s) to the <u>task</u> you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Input

Name	Туре	Description			
taskHandle	TaskHandle	The task used in this function.	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to us specify a <u>list or range</u> of physical channels.	se to create	e virtual channels. Yo	
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you mus names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> : the virtual channels.			
minVal	float64	The minimum value, in <b>units</b> , that you expe	ect to meas	ure.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expe	ect to meas	sure.	
units	int32	The units to use to return linear position	measurem	ents from the channe	
		Name		Description	
		DAQmx_Val_Meters		Meters	
		DAQmx_Val_Inches		Inches	
		DAQmx_Val_FromCustomScale Units scale custo to sp scale		Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
voltageExcitSource	int32	The source of excitation.			
		Value	Descri	ption	
		DAQmx_Val_Internal DAQmx_Val_External	Use the source must us to spec excitati Use an other th	e built-in excita of the device. ` se <b>voltageExci</b> ify the amount ion. excitation soundary the built-in	
			excitati device. <b>voltage</b>	ion source of the You must use <b>ExcitVal</b> to sp	

## DAQmx\_Val\_None

the amount of excitati

Supply no excitation t channel. You cannot u this value if the senso requires excitation.

voltageExcitVal	float64	The amount of excitation, in volts, that the sensor requires.		
voltageExcitFreq	float64	The excitation frequency, in hertz, that the sensor requires. Refer to the documentation to determine this value.		
ACExcitWireMode	int32	The number of leads on the sensor. Some sensors may require you to tie together to create a 4-wire or 5-wire sensor. Refer to the documentation sensor for more information.		
		Value Description		
		DAQmx_Val_4Wire	4-wire.	
		DAQmx_Val_5Wire 5-wire.		
customScaleName	const char []	[] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set <b>units</b> to DAQmx_Val_FromCustomScale. If you do not set <b>un</b> DAQmx_Val_FromCustomScale, you must set <b>customScaleName</b> to N		

#### Name Type Description

# DAQmxCreateTEDSAIPosRVDTChan

int32 DAQmxCreateTEDSAIPosRVDTChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 voltageExcitSource, float64 voltageExcitVal, float64 voltageExcitFreq, int32 ACExcitWireMode, const char customScaleName[]);

Creates channel(s) that use an  $_{\text{RVDT}}$  to  $_{\text{measure angular position}}$  and adds the channel(s) to the  $_{\text{task}}$  you specify with **taskHandle**. You must configure the physical channel(s) with TEDS information to use this function.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
physicalChannel	const char []	The names of the physical channels to u specify a <u>list or range</u> of physical channels	ise to create	e virtual channels. Yo
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you exp	ect to meas	ure.
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you exp	ect to meas	sure.
units	int32	The units to use to return angular position	on measure	ments from the chanı
		Name		Description
		DAQmx_Val_Degrees		Degrees
		DAQmx_Val_Radians Radia		
		DAQmx_Val_FromCustomScale Units spe a custom Use <b>customSc</b> to specify scale.		
voltageExcitSource	int32	The source of excitation.		
		Value	Descri	ption
		DAQmx_Val_Internal	Use the source must us to spec excitation	e built-in excita of the device. ` se <b>voltageExci</b> ify the amount ion.
		DAQmx_Val_External	Use an other th excitati device.	excitation sou nan the built-in ion source of th You must use

### **voltageExcitVal** to sp the amount of excitati

### DAQmx\_Val\_None

Supply no excitation 1 channel. You cannot u this value if the senso

requires excitation.

voltageExcitVal	float64	The amount of excitation, in volts, that the sensor requires.		
voltageExcitFreq	float64	The excitation frequency, in hertz, that the sensor requires. Refer to the documentation to determine this value.		
ACExcitWireMode	int32	The number of leads on the sensor. Some sensors may require you to the together to create a 4-wire or 5-wire sensor. Refer to the documentation sensor for more information.		
		Value	Description	
		DAQmx_Val_4Wire	4-wire.	
		DAQmx_Val_5Wire	5-wire.	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to must set <b>units</b> to DAQmx_Val_From DAQmx_Val_FromCustomScale, y	o the channel. To use this parameter, mCustomScale. If you do not set <b>un</b> i ou must set <b>customScaleName</b> to N	

#### Name Type Description

# DAQmxCreateAOCurrentChan

int32 DAQmxCreateAOCurrentChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, const char customScaleName[]);

Creates channel(s) to generate current and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description	
taskHandle	TaskHandle	The task to which to add the channels that this function creates.	
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.	
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> : the virtual channels.	
minVal	float64	The minimum value, in <b>units</b> , that you expect to generate.	
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you expect to generate.	
units	int32	The units in which to generate current.	
		Value	Description
		DAQmx_Val_Amps	amperes
		DAQmx_Val_FromCustomScale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set <b>units</b> to DAQmx_Val_FromCustomScale. If you do not set <b>uni</b>	

DAQmx\_Val\_FromCustomScale, you must set customScaleName to N

#### Name Type Description
# DAQmxCreateAOFuncGenChan

int32 DAQmxCreateAOFuncGenChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], int32 type, float64 frequency, float64 amplitude, float64 offset);

Creates a channel for continually generating a waveform on the selected physical channel.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	[] The name(s) to assign to the created virtual channel(s). If you do r specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you ref to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this functio you can specify a list of names separated by commas. If you provi fewer names than the number of virtual channels you create, NI-DAQmx automatically assigns names to the virtual channels.		
type	int32	Specifies the kind of waveform to genera	te.	
		Value Description		
		DAQmx_Val_Sine	Sine wave	
		DAQmx_Val_Triangle	Triangle wave	
		DAQmx_Val_Square	Sawtooth wave	
		DAQmx_Val_Sawtooth	Square wave	
frequency	float64	The frequency of the waveform to genera	te in hertz.	
amplitude	float64	The zero-to-peak amplitude of the waveform to generate in volts. Zero and negative values are valid.		
offset	float64	The voltage offset of the waveform to get	nerate.	

#### Name Type Description

# DAQmxCreateAOVoltageChan

int32 DAQmxCreateAOVoltageChan (TaskHandle taskHandle, const char physicalChannel[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, const char customScaleName[]);

Creates channel(s) to generate voltage and adds the channel(s) to the task you specify with **taskHandle**.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
physicalChannel	const char []	The names of the physical channels to use to create virtual channels. Yo specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> the virtual channels.		
minVal	float64	The <i>minimum value</i> , in <b>units</b> , that you expect to generate.		
maxVal	float64	The maximum value, in <b>units</b> , that you expect to generate.		
units	int32	The units in which to generate voltage.		
		Name	Description	
		DAQmx_Val_Volts	volts	
		DAQmx_Val_FromCustomScale Units a custo scale specifie customScale to specify a o scale.		
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. T	o use this parameter,	

must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **uni** DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to N

#### Name Type Description

# DAQmxCreateDIChan

int32 DAQmxCreateDIChan (TaskHandle taskHandle, const char lines[], const char nameToAssignToLines[], int32 lineGrouping);

Creates channel(s) to measure digital signals and adds the channel(s) to the task you specify with **taskHandle**. You can group digital lines into one digital channel or separate them into multiple digital channels. If you specify one or more entire ports in **lines** by using port physical channel names, you cannot separate the ports into multiple channels. To separate ports into multiple channels, use this function multiple times with a different port each time.

Input
-------

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
lines	const char []	The names of the digital lines used to create a virtual channel. You can specify a $list$ or range of lines.		
nameToAssignToLines	const char [] int32	The name of the created virtual channel(s). If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToLines</b> , you must use the names when you refer to these channels in other NI-DAQmx functions.		
1 3		channels. If you specify one or more entire ports in <b>lines</b> , you must set <b>lineGrouping</b> to DAQmx_Val_ChanForAllLines.		
		Value	Description	
		DAQmx_Val_ChanPerLine	One channel for each line	
		DAQmx_Val_ChanForAllLines	One channel for all lines	

#### Name Type Description

# DAQmxCreateDOChan

int32 DAQmxCreateDOChan (TaskHandle taskHandle, const char lines[], const char nameToAssignToLines[], int32 lineGrouping);

Creates channel(s) to generate digital signals and adds the channel(s) to the task you specify with **taskHandle**. You can group digital <u>lines</u> into one <u>digital channel</u> or separate them into multiple digital channels. If you specify one or more entire <u>ports</u> in **lines** by using port physical channel names, you cannot separate the ports into multiple channels. To separate ports into multiple channels, use this function multiple times with a different port each time.

Input
-------

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
lines	const char []	The names of the digital lines used to create a virtual channel. You can specify a $list$ or range of lines.		
nameToAssignToLines	const char [] int32	The name of the created virtual channel(s). If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToLines</b> , you must use the names when you refer to these channels in other NI-DAQmx functions.		
1 3		channels. If you specify one or more entire ports in <b>lines</b> , you must set <b>lineGrouping</b> to DAQmx_Val_ChanForAllLines.		
		Value	Description	
		DAQmx_Val_ChanPerLine	One channel for each line	
		DAQmx_Val_ChanForAllLines	One channel for all lines	

#### Name Type Description

# DAQmxCreateCICountEdgesChan

int32 DAQmxCreateCICountEdgesChan (TaskHandle taskHandle, const char counter[], const char nameToAssignToChannel[], int32 edge, uInt32 initialCount, int32 countDirection);

Creates a channel to count the number of rising or falling edges of a digital signal and adds the channel to the task you specify with **taskHandle**. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the <u>default input terminal</u> of the counter unless you select a different input terminal.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do no specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
edge	int32	Specifies on which edges of the input s decrement the count.	signal to ir	ncrement or
		Value	Descri	ption
		DAQmx_Val_Rising	Rising	edge(s).
		DAQmx_Val_Falling	Falling	gedge(s).
initialCount	uInt32	The value from which to start counting.		
countDirection	int32	Specifies whether to increment or decredge.	rement the	counter on each
		Value		Description
		DAQmx_Val_CountUp		Increment the count register on each edge.
	DAQmx_Val_CountDown		n	Decrement the count register on each edge.
DAQmx_Val_ExtControlled		The state of a digital line controls the count direction. Each counter has a default count		

direction terminal.

#### Name Type Description

# DAQmxCreateCIFreqChan

Creates a channel to measure the frequency of a digital signal and adds the channel to the task you specify with **taskHandle**. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channel If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you ex	pect to mea	sure.
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you ex	spect to mea	sure.
units	int32	The units to use to return the measured	ment.	
		Name		Description
		DAQmx_Val_Hz hertz DAQmx_Val_Ticks Timeb DAQmx_Val_FromCustomScale Custor to spec scale.		hertz
				Timebase tick
				Units by a cus scale. Use <b>customScale!</b> to specify a cu scale.
edge	int32	Specifies between which edges to mea	sure the free	quency or period of th
		Value	Descrip	otion
		DAQmx_Val_Rising	Rising e	edge(s).
		DAQmx_Val_Falling	Falling	edge(s).
measMethod	int32	The method used to calculate the period	od or freque	ncy of the signal.
		Value	]	Description
	DAQmx_Val_LowFreq1Ctr		Use one counter uses a constant timebase to mea the input signal.	
		DAQmx_Val_HighFreq2Ctr Use two		Use two counte

			one of which co pulses of the sig to measure during the specified measurement time
		DAQmx_Val_LargeRng2Ctr	Use one counter divide the frequ of the input sigr create a lower- frequency signa the second coun can more easily measure.
measTime	float64	The length of time to measure the frequency or p <b>measMethod</b> is DAQmx_Val_HighFreq2Ctr. M with increased measurement time and with incre	period of a digital signal leasurement accuracy in eased signal frequency.
		<b>Caution</b> If you measure a h for too long a time, the cour over, resulting in an incorre	igh-frequency sint register could in course of the second se
divisor	uInt32	The value by which to divide the input signal, w DAQmx_Val_LargeRng2Ctr. The larger this val measurement, but too large a value can cause the resulting in an incorrect measurement.	then <b>measMethod</b> is ue, the more accurate the e count register to roll o
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel must set <b>units</b> to DAQmx_Val_FromCustomSca DAQmx_Val_FromCustomScale, you must set	l. To use this parameter, ale. If you do not set <b>un</b> i c <b>ustomScaleName</b> to N

#### Name Type Description

# DAQmxCreateCIPeriodChan

Creates a channel to measure the period of a digital signal and adds the channel to the task you specify with **taskHandle**. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.



**Note** When **measMethod** is set to DAQmx\_Val\_LowFreq1Ctr, you must pass the values 0.000001 for **measTime** and 4 for **divisor**. These values will be ignored by the function, but if they are not passed, the function returns an error.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you ex	pect to mea	sure.
maxVal	float64	The maximum value, in <b>units</b> , that you ex	spect to mea	sure.
units	int32	The units to use to return the measurer	nent.	
		Value		Description
		DAQmx_Val_Seconds seconds DAQmx_Val_Ticks Time DAQmx_Val_FromCustomScale Units scale <b>cust</b> to sp scale		seconds
				Timebase tick
				Units a custon scale specifies <b>customScale</b> to specify a cu scale.
edge	int32	Specifies between which edges to mea	sure the free	quency or period of th
		Value	Descrip	otion
		DAQmx_Val_Rising	Rising e	edge(s).
		DAQmx_Val_Falling	Falling	edge(s).
measMethod	int32	Specifies the method used to calculate	the frequen	cy or period of the sig
	Value DAQmx_Val_LowFreq1Ctr   DAQmx_Val_LowFreq1Ctr 1   DAQmx_Val_HighFreq2Ctr 1		]	Description
			Ctr I i t t	Use one counter uses a constant timebase to mea the input signal.
			Use two counte	

			one of which co pulses of the sig to measure during the specified measurement time
		DAQmx_Val_LargeRng2Ctr	Use one counter divide the frequ of the input sigr create a lower- frequency signa the second coun can more easily measure.
measTime	float64	The length of time to measure the frequency or p <b>measMethod</b> is DAQmx_Val_HighFreq2Ctr. M with increased measurement time and with incre	period of a digital signal leasurement accuracy in eased signal frequency.
		<b>Caution</b> If you measure a h for too long a time, the cour over, resulting in an incorre	igh-frequency sint register could in course of the second se
divisor	uInt32	The value by which to divide the input signal, w DAQmx_Val_LargeRng2Ctr. The larger this val measurement, but too large a value can cause the resulting in an incorrect measurement.	then <b>measMethod</b> is ue, the more accurate the e count register to roll o
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel must set <b>units</b> to DAQmx_Val_FromCustomSca DAQmx_Val_FromCustomScale, you must set	l. To use this parameter, ale. If you do not set <b>un</b> i c <b>ustomScaleName</b> to N

#### Name Type Description

# DAQmxCreateCIPulseWidthChan

Creates a channel to measure the width of a digital pulse and adds the channel to the task you specify with **taskHandle**. **startingEdge** determines whether to measure a high pulse or a low pulse. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not sp name, NI-DAQmx uses the physical channel name as the virtual chann If you specify your own names for <b>nameToAssignToChannel</b> , you mu names when you refer to these channels in other NI-DAQmx functions If you create multiple virtual channels with one call to this function, yo specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assign</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you ex	pect to meas	ure.
maxVal	float64	The <u>maximum value</u> , in <b>units</b> , that you ex	pect to meas	ure.
units	int32	The units to use to return the measurement.		
		Value		Description
		DAQmx_Val_Seconds	seconds	
		DAQmx_Val_Ticks	Timebase tick	
		DAQmx_Val_FromCustomScale Uni scal <b>cus</b> to s scal		Units a custon scale specifies <b>customScale</b> to specify a cu scale.
startingEdge	int32	Specifies on which edge to begin mea	suring pulse	width.
		Value	Descrip	tion
		DAQmx_Val_Rising	Rising e	dge(s).
		DAQmx_Val_Falling Falling edg		edge(s).
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the must set <b>units</b> to DAQmx_Val_From DAQmx_Val_FromCustomScale, you	ne channel. T CustomScale. must set <b>cus</b>	o use this parameter, If you do not set <b>un</b> i <b>tomScaleName</b> to N

#### Name Type Description

# DAQmxCreateCISemiPeriodChan

int32 DAQmxCreateCISemiPeriodChan (TaskHandle taskHandle, const char counter[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, const char customScaleName[]);
Creates a channel to measure the time between state transitions of a digital signal and adds the channel to the task you specify with **taskHandle**. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the <u>default input terminal</u> of the counter unless you select a different input terminal.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not spename, NI-DAQmx uses the physical channel name as the virtual channe If you specify your own names for <b>nameToAssignToChannel</b> , you must names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
minVal	float64	The minimum value, in <b>units</b> , that you expect to measure.		
maxVal	float64	The maximum value, in <b>units</b> , that you expect to measure.		
units	int32	The units to use to return the measurement.		
		Value Description		
		DAQmx_Val_Seconds	seconds	
		DAQmx_Val_Ticks	Timebase tick	
		DAQmx_Val_FromCustomScale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. T	o use this parameter,	

const char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **un**i DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to N

### Name Type Description

# DAQmxCreateCITwoEdgeSepChan

DAQmxCreateCITwoEdgeSepChan (TaskHandle taskHandle, const char counter[], const char nameToAssignToChannel[], float64 minVal, float64 maxVal, int32 units, int32 firstEdge, int32 secondEdge, const char customScaleName[]);

Creates a channel that measures the amount of time between the rising or falling edge of one digital signal and the rising or falling edge of another digital signal. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the default input terminals of the counter unless you select different input terminals.

Name	Туре	Description	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.			
counter	const char []	The name of the counter to use to create virtual channels.			
nameToAssignToChannel	const char []	The name to assign to the created virtual channel. If you specify your ov for <b>nameToAssignToChannel</b> , you must use the names when you refer channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.			
minVal	float64	The minimum value, in <b>units</b> , that you ex	spect to meas	ure.	
maxVal	float64	The maximum value, in <b>units</b> , that you ex	spect to meas	sure.	
units	int32	The units to use to return the measured	ment.		
		Value		Description	
		DAQmx_Val_Seconds		seconds	
		DAQmx_Val_Ticks		Timebase tick	
		DAQmx_Val_FromCustomScale Units scale <b>custo</b> to spe scale.		Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
firstEdge	int32	Specifies on which edge of the first sig	gnal to start e	each measurement.	
		Name	Descrip	tion	
		DAQmx_Val_Rising	Start eac the risin signal.	ch measuremen g edge of the fi	
		DAQmx_Val_Falling	Start eac the fallin signal.	ch measuremen ng edge of the 1	
secondEdge	int32	Specifies on which edge of the first sig	gnal to stop e	each measurement.	
		Name	Descrip	tion	
		DAQmx_Val_Rising	Stop eac the risin	h measuremen g edge of the	

### second signal.

### DAQmx\_Val\_Falling

Stop each measuremen the falling edge of the second signal.

customScaleName

const char [] The name of a <u>custom scale</u> to apply to the channel. To use this parameter, must set **units** to DAQmx\_Val\_FromCustomScale. If you do not set **uni** DAQmx\_Val\_FromCustomScale, you must set **customScaleName** to N

### Name Type Description

# DAQmxCreateCILinEncoderChan

Creates a channel that uses a <u>linear encoder</u> to measure linear position. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the <u>default input terminals</u> of the counter unless you select different input terminals.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name to assign to the created virtual channel. If you specify your ov for <b>nameToAssignToChannel</b> , you must use the names when you refer channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
decodingType	int32	Specifies how to count and interpret the pulses that the encoder generate 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 only for two-pulse encoders. DAQmx_Val_X2 and DAQmx_Val_X4 decoding are more sensitive to changes in position than DAQmx_Val_X1 encoding, with DAQmx_Val being the most sensitive. However, more sensitive decoding is more like produce erroneous measurements if there is vibration in the encoder or onoise in the signals.		
		Value	Descriptio	
		DAQmx_Val_X1	If signal A signal B, co the rising e of signal A signal B lea signal A, co the falling edges of sig A.	
		DAQmx_Val_X2	Count the r and falling edges of si <sub>i</sub> A.	
		DAQmx_Val_X4	Count the r and falling edges of bc signal A an signal B.	

### DAQmx\_Val\_TwoPulseCounting Increment i count on ris edges of sig A. Decrem the count o rising pulse signal B.

ZidxEnable	bool32	Specifies whether to enable $\underline{z \text{ indexing}}$ for the me	easurement.	
ZidxVal	float64	The value, in <b>units</b> , to which to reset the measurement when signal Z is signal A and signal B are at the states you specify with <b>ZidxPhase</b> .		
ZidxPhase	int32	The states at which signal A and signal B must be while signal Z is high DAQmx to reset the measurement. If signal Z is never high while the sia and signal B are high, for example, you must choose a phase other than DAQmx_Val_AHighBHigh. When signal Z goes high and how long it stays high varies from encode encoder. Refer to the documentation for the encoder to determine the tim signal Z with respect to signal A and signal B.		
		Value	Description	
		DAQmx_Val_AHighBHigh	Reset the measurement wh both signal A and signal B are at hi logic.	
		DAQmx_Val_AHighBLow	Reset the measurement wh signal A is at hig logic and signal at low logic.	
		DAQmx_Val_ALowBHigh	Reset the measurement wh signal A is at lov logic and signal at high logic.	
		DAQmx_Val_ALowBLow	Reset the measurement wh both signal A and signal B are at lo logic.	

units	int32	The units to use to return linear position measurements from the char		
		Name	Description	
		DAQmx_Val_Meters	Meters.	
		DAQmx_Val_Inches	Inches.	
		DAQmx_Val_Ticks	Timebase Ticl	
		DAQmx_Val_FromCustomScale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
distPerPulse	float64	The distance measured for each pulse the encoder in <b>units</b> .	generates. Specify thi	
initialPos	float64	The position of the encoder when the measuremen <b>units</b> .	t begins. This value is	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. T must set <b>units</b> to DAQmx_Val_FromCustomScale DAQmx_Val_FromCustomScale, you must set <b>cus</b>	Fo use this parameter, . If you do not set <b>uni</b> stomScaleName to N	

### Name Type Description

# DAQmxCreateCIAngEncoderChan

Creates a channel that uses an <u>angular encoder</u> to measure angular position. You can create only one counter input channel at a time with this function because a task can include only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the <u>default input terminals</u> of the counter unless you select different input terminals.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels.		
nameToAssignToChannel	const char []	The name to assign to the created virtual channel. If you specify your ov for <b>nameToAssignToChannel</b> , you must use the names when you refer channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you specify a list of names separated by commas. If you provide fewer name the number of virtual channels you create, NI-DAQmx <u>automatically assigns</u> the virtual channels.		
decodingType	int32	Specifies how to count and interpret the pulses that the encoder generate 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 only for two-pulse encoders. DAQmx_Val_X2 and DAQmx_Val_X4 decoding are more sensitive to changes in position than DAQmx_Val_X1 encoding, with DAQmx_Val being the most sensitive. However, more sensitive decoding is more like produce erroneous measurements if there is vibration in the encoder or onoise in the signals.		
		Value	Descriptio	
		DAQmx_Val_X1	If signal A signal B, co the rising e of signal A signal B lea signal A, co the falling edges of sig A.	
		DAQmx_Val_X2	Count the r and falling edges of si <sub>i</sub> A.	
		DAQmx_Val_X4	Count the r and falling edges of bc signal A an signal B.	

### DAQmx\_Val\_TwoPulseCounting Increment i count on ris edges of sig A. Decrem the count o rising pulse signal B.

ZidxEnable	bool32	Specifies whether to enable $\underline{z \text{ indexing}}$ for the me	easurement.	
ZidxVal	float64	The value, in <b>units</b> , to which to reset the measurement when signal Z is signal A and signal B are at the states you specify with <b>ZidxPhase</b> .		
ZidxPhase	int32	The states at which signal A and signal B must be while signal Z is high DAQmx to reset the measurement. If signal Z is never high while the sia and signal B are high, for example, you must choose a phase other than DAQmx_Val_AHighBHigh. When signal Z goes high and how long it stays high varies from encode encoder. Refer to the documentation for the encoder to determine the tim signal Z with respect to signal A and signal B.		
		Value	Description	
		DAQmx_Val_AHighBHigh	Reset the measurement wh both signal A and signal B are at hi logic.	
		DAQmx_Val_AHighBLow	Reset the measurement wh signal A is at hig logic and signal at low logic.	
		DAQmx_Val_ALowBHigh	Reset the measurement wh signal A is at lov logic and signal at high logic.	
		DAQmx_Val_ALowBLow	Reset the measurement wh both signal A and signal B are at lo logic.	

units	int32	The units to use to return angular position measurements from the channel		
		Value	Description	
		DAQmx_Val_Degrees	Degrees	
		DAQmx_Val_Radians	Radians	
		DAQmx_Val_Ticks	Timebase tick	
		DAQmx_Val_FromCustomScale	Units a custon scale specifies <b>customScale</b> to specify a cu scale.	
pulsesPerRev	uInt32	The number of pulses the encoder generates per re- number of pulses on one of either A signal or B sig pulses on both signal A and signal B.	volution. This value is anal, not the total num	
initialAngle	float64	The starting angle of the encoder when the measur value in <b>units</b> .	ement begins. Specify	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to the channel. T must set <b>units</b> to DAQmx_Val_FromCustomScale DAQmx_Val_FromCustomScale, you must set <b>cus</b>	Co use this parameter, . If you do not set <b>uni</b> stomScaleName to N	

### Name Type Description

# DAQmxCreateCIGPSTimestampChan

Creates a channel that uses a special-purpose counter to take a timestamp and synchronizes that counter to a GPS receiver.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create	ate virtual cha	annels.
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx <u>automatically assigns names</u> to the virtual channels.		
units	int32	The units to use to return the timestamp.		
		Value Des		
		DAQmx_Val_Seconds	Seconds	
		DAQmx_Val_FromCusto	From custom scale	
gpsSyncMethod	int32	The method to use to synchronize the	counter to a	GPS receiver.
		Value	Descript	tion
		DAQmx_Val_IRIGB	IRIG-B	
		DAQmx_Val_PPS	DAQmx_Val_PPS PPS	
		DAQmx_Val_None	None	
customScaleName	const char []	The name of a <u>custom scale</u> to apply to t parameter, you must set <b>units</b> to DAC you do not set <b>units</b> to DAQmx_Val_ <b>customScaleName</b> to NULL.	he channel. T Qmx_Val_From FromCustom	'o use this mCustomScale. If Scale, you must set

### Name Type Description

# DAQmxCreateCOPulseChanFreq

int32 DAQmxCreateCOPulseChanFreq (TaskHandle taskHandle, const char counter[], const char nameToAssignToChannel[], int32 units, int32 idleState, float64 initialDelay, float64 freq, float64 dutyCycle);

Creates channel(s) to generate digital pulses that **freq** and **dutyCycle** define and adds the channel to the task you specify with **taskHandle**. The pulses appear on the default output terminal of the counter unless you select a different output terminal.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx automatically assigns names to the virtual channels.		
units	int32	The units in which to specify <b>freq</b> .		
		Name	Description	
		DAQmx_Val_Hz	hertz	
idleState	int32	The resting state of the output te	rminal.	
		Value	Description	
		DAQmx_Val_High	High state.	
		DAQmx_Val_Low	Low state.	
initialDelay	float64	The amount of time in seconds to wait before generating the first pulse.		
freq	float64	The frequency at which to gener	ate pulses.	
dutyCycle	float64	The width of the pulse divided by the pulse period. NI-DAQmx uses this ratio, combined with frequency, to determine pulse width and the interval between pulses.		

### Name Type Description

# DAQmxCreateCOPulseChanTicks

Creates channel(s) to generate digital pulses defined by the number of timebase ticks that the pulse is at a high state and the number of timebase ticks that the pulse is at a low state and also adds the channel to the task you specify with **taskHandle**. The pulses appear on the default output terminal of the counter unless you select a different output terminal.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI- DAQmx <u>automatically assigns names</u> to the virtual channels.		
sourceTerminal	const char []	The <u>terminal</u> to which you connect an external timebase. You also can specify a source terminal by using a <u>terminal name</u> .		
idleState	int32	The resting state of the output terminal.		
		Value	Description	
		DAQmx_Val_High	High state.	
		DAQmx_Val_Low	Low state.	
initialDelay	int32	The number of timebase ticks to w	ait before generating the first pulse.	
lowTicks	int32	The number of timebase ticks that the pulse is low.		
highTicks	int32	The number of timebase ticks that the pulse is high.		

### Name Type Description

# DAQmxCreateCOPulseChanTime

int32 DAQmxCreateCOPulseChanTime (TaskHandle taskHandle, const char counter[], const char nameToAssignToChannel[], int32 units, int32 idleState, float64 initialDelay, float64 lowTime, float64 highTime);

Creates channel(s) to generate digital pulses defined by the amount of time the pulse is at a high state and the amount of time the pulse is at a low state and adds the channel to the task you specify with **taskHandle**. The pulses appear on the default output terminal of the counter unless you select a different output terminal.

Name	Туре	Description		
taskHandle	TaskHandle	The task to which to add the channels that this function creates.		
counter	const char []	The name of the counter to use to create virtual channels. You can specify a <u>list or range</u> of physical channels.		
nameToAssignToChannel	const char []	The name(s) to assign to the created virtual channel(s). If you do not specify a name, NI-DAQmx uses the physical channel name as the virtual channel name. If you specify your own names for <b>nameToAssignToChannel</b> , you must use the names when you refer to these channels in other NI-DAQmx functions. If you create multiple virtual channels with one call to this function, you can specify a list of names separated by commas. If you provide fewer names than the number of virtual channels you create, NI-DAQmx automatically assigns names to the virtual channels.		
units	int32	The units in which to define pulse high and low time.		
		Value	Description	
		DAQmx_Val_Seconds	Seconds	
idleState	int32	The resting state of the output term	inal.	
		Value	Description	
		DAQmx_Val_High	High state.	
		DAQmx_Val_Low	Low state.	
initialDelay	float64	The amount of time in seconds to wait before generating the first pulse.		
lowTime	float64	The amount of time the pulse is low, in seconds.		
highTime	float64	The amount of time the pulse is high, in seconds.		

### Name Type Description
# DAQmxCfgBurstHandshakingTimingExportClock

int32 DAQmxCfgBurstHandshakingTimingExportClock (TaskHandle taskHandle, int32 sampleMode, uInt64 sampsPerChan, float64 sampleClkRate, const char sampleClkOutpTerm[], int32 sampleClkPulsePolarity, int32 pauseWhen, int32 readyEventActiveLevel);

## Purpose

Configures when the DAQ device transfers data to a peripheral device, using the DAQ device's onboard sample clock to control <u>burst handshaking timing</u>.

### **Parameters**

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
sampleMode	int32	Specifies whether the task acquires or generates samples continuously or i acquires or generates a finite number of samples.		
		Value	Description	
		DAQmx_Val_FiniteSamps	Acquire or generate a finite number of samples.	
		DAQmx_Val_ContSamps	Acquire or generate samples until you stop the task.	
		DAQmx_Val_HWTimedSinglePoint	Acquire or generate samples continuousl using hardware timing without a buffer. <u>Hardwar</u> timed single point sample mode is supported only for the sample clock and change detection timing	

### types.

sampsPerChan	uInt64	The number of samples to acquire from each channel if <b>sampleMode</b> is DAQmx_Val_FiniteSamps. If <b>sampleMode</b> is DAQmx_Val_ContSamps, NI-DAQmx uses this value to <u>determine the buffer size</u> .
sampleClkRate	float64	Specifies the sampling rate in samples per channel per second. If you use a external source for the Sample Clock, set this input to the maximum expected rate of that clock.
sampleClkOutpTerm	const char []	Specifies the terminal to which to route the Sample Clock.
sampleClkPulsePolarity	int32	Specifies if the polarity for the exported sample clock is active high or act low.
pauseWhen	int32	Specifies whether the task pauses while the signal is high or low.
readyEventActiveLevel	int32	Specifies the polarity for the Ready for Transfer event.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgBurstHandshakingTimingImportClock

int32 DAQmxCfgBurstHandshakingTimingImportClock (TaskHandle taskHandle, int32 sampleMode, uInt64 sampsPerChan, float64 sampleClkRate, const char sampleClkSrc[], int32 sampleClkActiveEdge, int32 pauseWhen, int32 readyEventActiveLevel);

## Purpose

Configures when the DAQ device transfers data to a peripheral device, using an imported sample clock to control <u>burst handshaking timing</u>.

### **Parameters**

Input					
Name	Туре	Description			
taskHandle	TaskHandle	The task used in this function.			
sampleMode	int32	Specifies whether the task acquires or generates samples continuously or if acquires or generates a finite number of samples.			
		Value	Description		
		DAQmx_Val_FiniteSamps	Acquire or generate a finite number of samples.		
		DAQmx_Val_ContSamps	Acquire or generate samples until you stop the task.		
		DAQmx_Val_HWTimedSinglePoint	Acquire or generate samples continuously using hardware timing without a buffer. <u>Hardwars</u> timed single point sample mode is supported only for the sample clock and change detection timing		

types.

sampsPerChan	uInt64	The number of samples to acquire from each channel if <b>sampleMode</b> is DAQmx_Val_FiniteSamps. If <b>sampleMode</b> is DAQmx_Val_ContSamps, NI-DAQmx uses this value to <u>determine the buffer size</u> .		
sampleClkRate	float64	Specifies the sampling rate in samples per channel per second. If you use a external source for the Sample Clock, set this input to the maximum expected rate of that clock.		
sampleClkSrc	const char []	Specifies the terminal of the signal to use as the Sample Clock.		
sampleClkActiveEdge	int32	Specifies on which edge of a clock pulse sampling takes place. This proper is useful primarily when the signal you use as the Sample Clock is not a periodic clock.		
		Value Description		
		DAQmx_Val_Rising	Acquire or generate samples on the rising edges of the Sample Clock.	
		DAQmx_Val_Falling	Acquire or generate samples on the falling edges of the Sample Clock.	
pauseWhen	int32	Specifies whether the task pauses while the signal is high or low.		

readyEventActiveLevel int32

Specifies the polarity for the Ready for Transfer event.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgChangeDetectionTiming

int32 DAQmxCfgChangeDetectionTiming (TaskHandle taskHandle, const char risingEdgeChan[], const char fallingEdgeChan[], int32 sampleMode, uInt64 sampsPerChan);

## Purpose

Configures the task to acquire samples on the rising and/or falling edges of the lines or ports you specify.

## Parameters

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
risingEdgeChan	const char []	The names of the digital lines or ports on which to detect rising edges. You can specify a list or range of channels.		
fallingEdgeChan	const char []	The names of the digital lines or ports on which to detect <u>specify a list or range of channels</u> .	falling edges. You can	
sampleMode	int32	Specifies whether the task acquires samples continuously or if it acquires a finite number of samples.		
		Value	Description	
		DAQmx_Val_FiniteSamps	Acquire or generate a finite number of samples.	
		DAQmx_Val_ContSamps	Acquire or generate samples until you stop the task.	
		DAQmx_Val_HWTimedSinglePoint	Acquire or generate samples continuously using hardware timing without a buffer. <u>Hardware</u> timed single point sample mode is supported only for the sample	

clock and change detection timing types.

sampsPerChan uInt64

The number of samples to acquire from each channel if **sampleMode** is DAQmx\_Val\_FiniteSamps. If **sampleMode** is DAQmx\_Val\_ContSamps, NI-DAQmx uses this value to <u>determine the buffer size</u>.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgHandshakingTiming

int32 DAQmxCfgHandshakingTiming (TaskHandle taskHandle, int32 sampleMode, uInt64 sampsPerChanToAcquire);

## Purpose

Determines the number of digital samples to acquire or generate using digital handshaking between the device and a peripheral device.

# Parameters

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
sampleMode	int32	Specifies whether the task acquires or generates samples continuously of acquires or generates a finite number of samples.		
		Value	Descriptio	
		DAQmx_Val_FiniteSamps	Acquire or generate a finite number of samples.	
		DAQmx_Val_ContSamps	Acquire or generate samples until you stop the task.	
		DAQmx_Val_HWTimedSinglePoint	Acquire or generate samples continuous using hardware timing without a buffer. <u>Hardw</u> imed single point sample mode is supported only for th sample clock and change detection timing	

types.

sampsPerChanToAcquire uInt64

The number of samples to acquire or generate for each channel in the tas **sampleMode** is DAQmx\_Val\_FiniteSamps. If **sampleMode** is DAQmx\_Val\_ContSamps, NI-DAQmx uses this value to <u>determine the buff</u> <u>size</u>.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgImplicitTiming

int32 DAQmxCfgImplicitTiming (TaskHandle taskHandle, int32 sampleMode, uInt64 sampsPerChanToAcquire);

## Purpose

Sets only the number of samples to acquire or generate without specifying timing. Typically, you should use this function when the task does not require sample timing, such as tasks that use counters for buffered frequency measurement, buffered period measurement, or pulse train generation.

# Parameters

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
sampleMode	int32	Specifies whether the task acquires or generates samples continuously of acquires or generates a finite number of samples.		
		Value	Descriptio	
		DAQmx_Val_FiniteSamps	Acquire or generate a finite number of samples.	
		DAQmx_Val_ContSamps	Acquire or generate samples until you stop the task.	
		DAQmx_Val_HWTimedSinglePoint	Acquire or generate samples continuous using hardware timing without a buffer. <u>Hardw</u> imed single point sample mode is supported only for th sample clock and change detection timing	

types.

sampsPerChanToAcquire uInt64

The number of samples to acquire or generate for each channel in the tas **sampleMode** is DAQmx\_Val\_FiniteSamps. If **sampleMode** is DAQmx\_Val\_ContSamps, NI-DAQmx uses this value to <u>determine the buff</u> <u>size</u>.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgPipelinedSampClkTiming

int32 DAQmxCfgPipelinedSampClkTiming (TaskHandle taskHandle, const char source[], float64 rate, int32 activeEdge, int32 sampleMode, uInt64 samplesPerChannel);

## Purpose

Sets the source of the Sample Clock, the rate of the Sample Clock, and the number of samples to acquire or generate. The device acquires or generates samples on each Sample Clock edge, but does not respond to certain triggers until a few Sample Clock edges later. Pipelining allows higher data transfer rates at the cost of increased trigger response latency. Refer to the device documentation for information about which triggers pipelining affects.

This timing type allows handshaking using the Pause trigger, the Ready for Transfer event, or the Data Active event. Refer to the device documentation for more information.

This timing type is supported only by the NI 6536 and NI 6537.

# Parameters

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
source	const char []	The source <u>terminal</u> of the Sample Clock. To use the internal clock of the device, use NULL or use OnboardClock.		
rate	float64	The <u>sampling rate</u> in samples per second per channel. If you use an exter source for the Sample Clock, set this value to the maximum expected that clock.		u use an externation expected ra
activeEdge	int32	Specifies on which edge of the clock t	k to acquire or generate samples.	
		Value	Description	
		DAQmx_Val_Rising	Acquire or generate samples on the rising edges of the Sample Clock.	
		DAQmx_Val_Falling	Acquire or generat samples on the fall edges of the Sampl Clock.	
sampleMode	int32	Specifies whether the task acquires or generates sample acquires or generates a finite number of samples.		s continuously o
		Value		Descriptio
		DAQmx_Val_FiniteSamp	)S	Acquire or generate a finite number of samples.
		DAQmx_Val_ContSamps	3	Acquire or generate samples until you stop the task.
		DAQmx_Val_HWTimedS	SinglePoint	Acquire or generate samples

continuou using hardware timing without a buffer. Hardy timed single point sample mode is supported only for th sample clock and change detection timing types.

#### sampsPerChanToAcquire uInt64

The number of samples to acquire or generate for each channel in the ta **sampleMode** is DAQmx\_Val\_FiniteSamps. If **sampleMode** is DAQmx\_Val\_ContSamps, NI-DAQmx uses this value to <u>determine the buf</u> <u>size</u>.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgSampClkTiming

int32 DAQmxCfgSampClkTiming (TaskHandle taskHandle, const char source[], float64 rate, int32 activeEdge, int32 sampleMode, uInt64 sampsPerChanToAcquire);

## Purpose

Sets the source of the  $_{\text{Sample Clock}}$ , the rate of the Sample Clock, and the number of samples to acquire or generate.

# Parameters

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
source	const char []	The source <u>terminal</u> of the Sample Clock. To use the internal clock of the device, use NULL or use OnboardClock.		
rate	float64	The <u>sampling rate</u> in samples per second per channel. If you use an exter source for the Sample Clock, set this value to the maximum expected that clock.		u use an externation expected ra
activeEdge	int32	Specifies on which edge of the clock t	k to acquire or generate samples.	
		Value	Description	
		DAQmx_Val_Rising	Acquire or generate samples on the rising edges of the Sample Clock.	
		DAQmx_Val_Falling	Acquire or generat samples on the fall edges of the Sampl Clock.	
sampleMode	int32	Specifies whether the task acquires or generates sample acquires or generates a finite number of samples.		s continuously o
		Value		Descriptio
		DAQmx_Val_FiniteSamp	)S	Acquire or generate a finite number of samples.
		DAQmx_Val_ContSamps	3	Acquire or generate samples until you stop the task.
		DAQmx_Val_HWTimedS	SinglePoint	Acquire or generate samples

continuou using hardware timing without a buffer. Hardy timed single point sample mode is supported only for th sample clock and change detection timing types.

#### sampsPerChanToAcquire uInt64

The number of samples to acquire or generate for each channel in the ta **sampleMode** is DAQmx\_Val\_FiniteSamps. If **sampleMode** is DAQmx\_Val\_ContSamps, NI-DAQmx uses this value to <u>determine the buf</u> <u>size</u>.

### **Return Value**

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgAnlgEdgeStartTrig

int32 DAQmxCfgAnlgEdgeStartTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerSlope, float64 triggerLevel);
Configures the task to start acquiring or generating samples when an analog signal crosses the level you specify.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
triggerSource	const char []	The name of a channel or <u>terminal</u> 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 first channel in the task. The only terminal you can use for E Series devices is PFI0.	
triggerSlope	int32	Specifies on which slope of the signal to start acquiring or generating samples when the signal crosses <b>triggerLevel</b> .	
		Value	Description
		DAQmx_Val_RisingSlope	Trigger on the rising slope of the signal.
		DAQmx_Val_FallingSlope	Trigger on the falling slope of the signal.
triggerLevel	float64	The threshold at which to start acquiring or ge in the units of the measurement or generation.	nerating samples. Specify this value Use <b>triggerSlope</b> to specify on

which slope to trigger at this threshold.

#### Name Type Description

# DAQmxCfgAnlgWindowStartTrig

int32 DAQmxCfgAnlgWindowStartTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerWhen, float64 windowTop, float64 windowBottom);

Configures the task to start acquiring or generating samples when an analog signal enters or leaves a range you specify.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
triggerSource	const char []	The name of a virtual channel or <u>terminal</u> where there is an analog signal to use as the source of the trigger. For E Series devices, if you use a virtual channel, it must be the first channel in the task. The only terminal you can use for E Series devices is PFI0.	
triggerWhen	int32	Specifies whether the task starts measuring or generating samples when the signal enters the window or when it leaves the window. Use <b>windowBottom</b> and <b>windowTop</b> to specify the limits of the window.	
	Value De		Description
		DAQmx_Val_EnteringWin	Trigger when the signal enters the window.
		DAQmx_Val_LeavingWin	Trigger when the signal leaves the window.
windowTop	float64	The upper limit of the window. Specify this va or generation.	alue in the units of the measurement

windowBottom float64 The lower limit of the window. Specify this value in the units of the measurement or generation.

#### Name Type Description

# DAQmxCfgDigEdgeStartTrig

int32 DAQmxCfgDigEdgeStartTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerEdge);

Configures the task to start acquiring or generating samples on a rising or falling edge of a digital signal.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
triggerSource	const char []	The name of a <u>terminal</u> where there is a digital signal to use as the source of the trigger.	
triggerEdge	int32	Specifies on which edge of a digital signal to start acquiring or generating samples.	
		Value	Description
		DAQmx_Val_Rising	Rising edge(s).
		DAQmx_Val_Falling	Falling edge(s).

#### Name Type Description

# DAQmxCfgDigPatternStartTrig

int32 DAQmxCfgDigPatternStartTrig (TaskHandle taskHandle, const char triggerSource[], const char triggerPattern[], int32 triggerWhen);

Configures a task to start acquiring or generating samples when a digital pattern is matched.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
triggerSource	const char []	Specifies the physical channels to use for pattern matching. physical channels determines the order of the pattern. If a per order of the physical channels within the port is in ascending	The order of the ort is included, the g order.
triggerPattern	const char []	Specifies the digital pattern that must be met for the trigger to e	occur.
triggerWhen	int32	Specifies the conditions under which the trigger occurs.	
		Value	Description
		DAQmx_Val_PatternMatches	Pattern matches
		DAQmx_Val_PatternDoesNotMatch	Pattern does not match

#### Name Type Description

# DAQmxDisableStartTrig

int32 DAQmxDisableStartTrig (TaskHandle taskHandle);

Configures the task to start acquiring or generating samples immediately upon starting the task.

InputNameTypeDescriptiontaskHandleTaskHandleThe task used in this function.

#### Name Type Description

# DAQmxCfgAnlgEdgeRefTrig

int32 DAQmxCfgAnlgEdgeRefTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerSlope, float64 triggerLevel, uInt32 pretriggerSamples);

Configures the task to stop the acquisition when the device acquires all pretrigger samples, an analog signal reaches the level you specify, and the device acquires all post-trigger samples.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
triggerSource	const char []	The name of a virtual channel or <u>terminal</u> where there is an analog signal to use as the source of the trigger. For E Series devices, if you use a virtual channel, it must be the only channel in the task. The only terminal you can use for E Series devices is PFI0.		
triggerSlope	int32	Specifies on which slope of the signal the Reference Trigger occurs.		
		Value	Description	
		DAQmx_Val_RisingSlope	Trigger on the rising slope of the signal.	
		DAQmx_Val_FallingSlope	Trigger on the falling slope of the signal.	
triggerLevel	float64	Specifies at what threshold to trigger. Specify measurement or generation. Use <b>triggerSlope</b> trigger at this threshold.	this value in the units of the to specify on which slope to	
pretriggerSamples	uInt32	The minimum number of samples per channel to acquire before recognizing the Reference Trigger. The number of posttrigger samples per channel is equal to number of samples per channel in the NI-DAQmx Timing functions minus <b>pretriggerSamples</b> .		

#### Name Type Description

# DAQmxCfgAnlgWindowRefTrig

int32 DAQmxCfgAnlgWindowRefTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerWhen, float64 windowTop, float64 windowBottom, uInt32 pretriggerSamples);

Configures the task to stop the acquisition when the device acquires all pretrigger samples, an analog signal enters or leaves a range you specify, and the device acquires all post-trigger samples.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
triggerSource	const char []	The name of a virtual channel or <u>terminal</u> where there is an analog signal to use as the source of the trigger. For E Series devices, if you use a virtual channel, it must be the only channel in the task. The only terminal you can use for E Series devices is PFI0.		
triggerWhen	int32	Specifies whether the Reference Trigger occurs when the signal enters the window or when it leaves the window. Use <b>windowBottom</b> and <b>windowTop</b> to specify the limits of the window.		
		Value	Description	
		DAQmx_Val_EnteringWin	Trigger when the signal enters the window.	
		DAQmx_Val_LeavingWin	Trigger when the signal leaves the window.	
windowTop	float64	The upper limit of the window. Specify this value in the units of the measurement or generation.		
windowBottom	float64	The lower limit of the window. Specify this value in the units of the measurement or generation.		
pretriggerSamples	uInt32	The minimum number of samples per channel to acquire before recognizing the Reference Trigger. The number of posttrigger samples per channel is equal to number of samples per channel in the NI-DAQmx Timing functions minus <b>pretriggerSamples</b> .		

#### Name Type Description

# DAQmxCfgDigEdgeRefTrig

int32 DAQmxCfgDigEdgeRefTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerEdge, uInt32 pretriggerSamples);

Configures the task to stop the acquisition when the device acquires all pretrigger samples, detects a rising or falling edge of a digital signal, and acquires all posttrigger samples.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
triggerSource	const char []	Specifies the name of a <u>terminal</u> where there is a digital signal to use as the source of the trigger.		
triggerEdge	int32	Specifies on which edge of the digital signal the Reference Trigger occurs.		
		Value	Description	
		DAQmx_Val_Rising	Rising edge(s).	
		DAQmx_Val_Falling	Falling edge(s).	
pretriggerSamples	uInt32	The minimum number of samples per channel to acquire before recognizing the Reference Trigger. The number of posttrigger samples per channel is equal to number of samples per channel in the NI-DAQmx Timing functions minus <b>pretriggerSamples</b> .		

#### Name Type Description

# DAQmxCfgDigPatternRefTrig

int32 DAQmxCfgDigPatternRefTrig (TaskHandle taskHandle, const char triggerSource[], const char triggerPattern[], int32 triggerWhen, uInt32 pretriggerSamples);

Configures the task to stop the acquisition when the device acquires all pretrigger samples, matches or does not match a digital pattern, and acquires all posttrigger samples.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
triggerSource	const char []	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.	
triggerPattern	const char []	Specifies the digital pattern that must be met for the trigger to occur.	
triggerWhen	int32	Specifies the conditions under which the trigger occurs.	
		Value	Description
		DAQmx_Val_PatternMatches	Pattern matches
		DAQmx_Val_PatternDoesNotMatch	Pattern does not match
pretriggerSamples	uInt32	The minimum number of samples per channel to acquire the Reference Trigger. The number of posttrigger samples equal to number of samples per channel in the NI-DAQm minus <b>pretriggerSamples</b> .	before recognizing s per channel is tx Timing functions

#### Name Type Description

# DAQmxDisableRefTrig

int32 DAQmxDisableRefTrig (TaskHandle taskHandle);
Disables reference triggering for the measurement or generation.

InputNameTypeDescriptiontaskHandleTaskHandleThe task used in this function.

#### Name Type Description

# DAQmxCfgDigEdgeAdvTrig

int32 DAQmxCfgDigEdgeAdvTrig (TaskHandle taskHandle, const char triggerSource[], int32 triggerEdge);

Configures a switch task to advance to the next entry in a scan list on a rising or falling edge of a digital signal.

InputNameTypeDescriptiontaskHandleTaskHandleThe task used in this function.triggerSourceconst char[]The terminal where there is a digital signal to use as the source of the trigger.triggerEdgeint32Specifies on which edge of a digital signal to advance to the next entry in the scan<br/>list.

Value	Description
DAQmx_Val_Rising	Rising edge(s).
DAQmx_Val_Falling	Falling edge(s).

#### Name Type Description

# DAQmxDisableAdvTrig

int32 DAQmxDisableAdvTrig (TaskHandle taskHandle);

Disables the advance triggering for the task.

InputNameTypeDescriptiontaskHandleTaskHandleThe task used in this function.

#### Name Type Description

# DAQmxSendSoftwareTrigger

int32 DAQmxSendSoftwareTrigger (TaskHandle taskHandle, int32 triggerID);

Generates the specified software trigger.

InputNameTypeDescriptiontaskHandleTaskHandleThe task used in this function.triggerIDint32Specifies which software trigger to generate.

Value

### Description

DAQmx\_Val\_AdvanceTrigger

Generate the advance trigger

#### Name Type Description

# DAQmxReadAnalogF64

Reads multiple floating-point samples from a task that contains one or more analog input channels.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task to read samples from.	
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.	
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.	
fillMode	bool32	Specifies whether or not the samples are <i>interleaved</i> .	
		Value	Description
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are	e read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.	
Output			
Name	Туре	Description	
readArray	float64 []	The array to read samples into, organized according to <b>fillMode</b> .	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.	

#### Name Type Description

# DAQmxReadAnalogScalarF64

int32 DAQmxReadAnalogScalarF64 (TaskHandle taskHandle, float64 timeout, float64 \*value, bool32 \*reserved);

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

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read the sample from.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
value	float64 *	The sample read from the task.

#### Name Type Description

# DAQmxReadBinaryI16

Reads multiple unscaled, signed 16-bit integer samples from a task that contains one or more analog input channels.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	e The task to read samples from.	
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.	
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.	
fillMode	<b>illMode</b> bool32 Specifies whether or not the samples are <u>interleaved</u> .		
		Value	Description
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.	
Output			
Name	Туре	Description	
readArray	int16 []	The array to read samples into, organized according to <b>fillMode</b> .	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.	

#### Name Type Description

# DAQmxReadBinaryI32

Reads one or more <u>unscaled</u> 32-bit signed integer samples from a task that contains one or more <u>analog input channels</u>.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	e The task used in this function.	
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.	
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.	
fillMode	<b>illMode</b> bool32 Specifies whether or not the samples are <u>interleaved</u> .		
		Value	Description
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.	
Output			
Name	Туре	Description	
readArray	int32 []	The array to read samples into, organized according to <b>fillMode</b> .	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.	

#### Name Type Description

# DAQmxReadBinaryU16

Reads multiple <u>unscaled</u>, unsigned 16-bit integer samples from a task that contains one or more analog input channels.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	e The task to read samples from.	
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.	
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.	
fillMode	bool32 Specifies whether or not the samples are <u>interleaved</u> .		
		Value	Description
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are	e read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.	
Output			
Name	Туре	Description	
readArray	uInt16 []	The array to read samples into, organized according to <b>fillMode</b> .	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.	

#### Name Type Description

# DAQmxReadBinaryU32
Reads one or more <u>unscaled</u> 32-bit unsigned integer samples from a task that contains one or more <u>analog input channels</u>.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task used in this function.		
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.		
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.		
fillMode	bool32	Specifies whether or not the samples are <i>interleaved</i> .		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)	
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)	
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are	e read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
readArray	uInt32 []	The array to read samples into, organized according to ${\bf f}$	illMode.	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.		

### Name Type Description

# DAQmxReadCounterF64

Reads multiple floating-point samples from a counter task. Use this function when counter samples are scaled to a floating-point value, such as for frequency and period measurements.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read samples from.
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
readArray	float64 []	The array to read samples into.
sampsPerChanRead	int32 *	The actual number of samples read from each channel.

### Name Type Description

# DAQmxReadCounterScalarF64

Reads a single floating-point sample from a counter task. Use this function when the counter sample is scaled to a floating-point value, such as for frequency and period measurement.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read the sample from.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
value	float64 *	The sample read from the task.

### Name Type Description

# DAQmxReadCounterScalarU32

int32 DAQmxReadCounterScalarU32 (TaskHandle taskHandle, float64 timeout, uInt32 \*value, bool32 \*reserved);

Reads a 32-bit integer sample from a counter task. Use this function when the counter sample is returned unscaled, such as for edge counting.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read the sample from.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
value	uInt32 *	The sample read from the task.

### Name Type Description

# DAQmxReadCounterU32

Reads multiple 32-bit integer samples from a counter task. Use this function when counter samples are returned unscaled, such as for edge counting.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read samples from.
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
readArray	uInt32 []	The array to read samples into.
sampsPerChanRead	int32 *	The actual number of samples read from each channel.

### Name Type Description

# DAQmxReadDigitalLines

Reads multiple samples from each digital line in a task. Each line in a channel gets one byte per sample.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task to read samples from.		
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Data property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.		
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.		
fillMode	bool32	Specifies whether or not the samples are <u>interleaved</u> .		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)	
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)	
arraySizeInBytes	uInt32	The size of the array, in samples, into which samples are	e read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
readArray	uInt8 []	The array to read samples into. Each <b>numBytesPerSam</b> one sample per channel, with each element in that group a line in that channel, up to the number of lines contained	<b>p</b> corresponds to bing corresponding to ed in the channel.	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.		

**numBytesPerSamp** int32 \* The number of elements in **readArray** that constitutes a sample per channel. For each sample per channel, **numBytesPerSamp** is the number of bytes that channel consists of.

### Name Type Description

# DAQmxReadDigitalScalarU32

int32 DAQmxReadDigitalScalarU32 (TaskHandle taskHandle, float64 timeout, uInt32 \*value, bool32 \*reserved);

Reads a single 32-bit integer sample from a task that contains a single digital input <u>channel</u>. Use this function for devices with up to 32 lines per port. The sample is returned in unsigned <u>integer format</u>.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read the sample from.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample. The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested sample. If the entire requested sample is read, the function is successful. Otherwise, the function returns a timeout error and returns what was actually read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
value	uInt32 *	The sample read from the task.

### Name Type Description

# DAQmxReadDigitalU8

int32 DAQmxReadDigitalU8 (TaskHandle taskHandle, int32 numSampsPerChan, float64 timeout, bool32 fillMode, uInt8 readArray[], uInt32 arraySizeInSamps, int32 \*sampsPerChanRead, bool32 \*reserved);

Reads multiple 8-bit integer samples from a task that has one or more multiple digital input channels. Use this function for devices with up to 8 lines per port. The samples are returned in unsigned byte format.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.		
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.		
fillMode	bool32	Specifies whether or not the samples are <i>interleaved</i> .		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)	
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)	
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are	e read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
readArray	uInt8 []	The array to read samples into, organized according to <b>f</b>	illMode.	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.		

### Name Type Description

# DAQmxReadDigitalU16

Reads multiple 16-bit integer samples from a task that contains one or more digital input channels. Use this function for devices with up to 16 lines per port. The samples are returned in unsigned integer format.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task to read samples from.		
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.		
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.		
fillMode	bool32	Specifies whether or not the samples are <i>interleaved</i> .		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)	
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)	
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are	e read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
readArray	uInt16 []	The array to read samples into, organized according to <b>f</b>	illMode.	
sampsPerChanRead	int32 *	The actual number of samples read from each channel.		

### Name Type Description
# DAQmxReadDigitalU32

Reads multiple 32-bit integer samples from a task that contains one or more digital input channels. Use this function for devices with up to 32 lines per port. The samples are returned in unsigned integer format.

Input				
Name	Туре	Description		
taskHandle	TaskHandle	The task to read samples from.		
numSampsPerChan	int32	The number of samples, per channel, to read. The default value of -1 (DAQmx_Val_Auto) reads all available samples. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Samples property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.		
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.		
fillMode	bool32	Specifies whether or not the samples are <i>interleaved</i> .		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (non- interleaved)	
		DAQmx_Val_GroupByScanNumber	Group by scan number (interleaved)	
arraySizeInSamps	uInt32	The size of the array, in samples, into which samples are	e read.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
readArray	uInt32 []	The array to read samples into, organized according to <b>fillMode</b> .		
sampsPerChanRead	int32 *	The actual number of samples read from each channel.		

#### Name Type Description

# DAQmxGetNthTaskReadChannel

Takes a task, an index, and a user declared buffer with size. It fills out the buffer with the *N*th channel as requested by the user in the index. This comes from all the channels in the task. Indexing starts at 1.

Input
-------

Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
index	uInt32	The <i>N</i> th channel to return. The <b>index</b> starts at 1.
bufferSize	int32	The size, in bytes, of <b>buffer</b> . If you pass 0, this function returns the number of bytes needed to allocate.
Output		
buffer	char []	The <i>N</i> th channel in the index. If you pass NULL, this function returns the number of bytes needed to allocate.

#### Name Type Description

status int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error. For this function, if you pass NULL for the buffer or 0 for the buffer size, this function returns the number of bytes needed to allocate.

# DAQmxReadRaw

int32 DAQmxReadRaw (TaskHandle taskHandle, int32 numSampsPerChan, float64 timeout, void \*readArray, uInt32 arraySizeInBytes, int32 \*sampsRead, int32 \*numBytesPerSamp, bool32 \*reserved);

Reads  $_{raw}$  samples directly from the input lines. There is no scaling, parsing, or separation of the samples. Refer to the specifications for your device to determine the format of the incoming samples.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to read samples from.
numSampsPerChan	int32	The number of samples, per channel, to read. If you set this parameter to -1 (DAQmx_Val_Auto), NI-DAQmx determines how many samples to read based on whether the task acquires samples continuously or acquires a finite number of samples. If the task acquires samples continuously and you set this parameter to -1, this function reads all the samples currently available in the buffer. If <b>readArray</b> does not contain enough space, this function returns as many samples as fit in <b>readArray</b> . If the task acquires a finite number of samples and you set this parameter to -1, the function waits for the task to acquire all requested samples, then reads those samples. If you set the Read All Available Data property to TRUE, the function reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.
timeout	float64	The amount of time, in seconds, to wait for the function to read the sample(s). The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to read the requested samples. If all the requested samples are read, the function is successful. Otherwise, the function returns a timeout error and returns the samples that were actually read.
arraySizeInBytes	uInt32	The size of the array into which samples are read.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
readArray	void *	The array into which samples are read.
sampsRead	int32 *	The actual number of bytes read into the array per scan.
numBytesPerSamp	int32 *	The number of elements in <b>readArray</b> that constitutes a sample. This value takes into account all channels that are read. For example, five channels giving 12 bytes for one entire scan return 12 bytes. The value is not divided along channel lines.

#### Name Type Description

# DAQmxWriteAnalogF64

Writes multiple floating-point samples to a task that contains one or more analog output channels.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of 2 samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this fir returns an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.		
dataLayout	bool32	Specifies how the samples are arranged, either $\underline{\text{interleaved}}$	or noninterleaved.	
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)	
writeArray	float64 []	The array of 64-bit samples to write to the task.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
sampsPerChanWritten	<b>ritten</b> int32 * The actual number of samples per channel successfully written to the buffe		written to the buffe	

#### Name Type Description

# DAQmxWriteAnalogScalarF64

int32 DAQmxWriteAnalogScalarF64 (TaskHandle taskHandle, bool32 autoStart, float64 timeout, float64 value, bool32 \*reserved);

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

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to write the sample to.
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do not start it.
timeout	float64	The amount of time, in seconds, to wait for this function to write the <b>value</b> . The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the value. If this function successfully writes the value, it does not return an error. Otherwise, the function returns a timeout error.
value	float64	A 64-bit sample to write to the task.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.

#### Name Type Description

# DAQmxWriteBinaryI16

Writes <u>unscaled</u> signed integer samples to the task.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of 2 samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description	
taskHandle	TaskHandle	The task to write samples to.	
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this fureturns an error.	
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do start it.	
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.	
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterleaved	
		Value	Description
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)
writeArray	int16 []	The array of 16-bit samples to write to the task.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.	
Output			
Name	Туре	Description	
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully written to the buffe	

#### Name Type Description

# DAQmxWriteBinaryU16

Writes <u>unscaled</u>, unsigned 16-bit integer samples to the task.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of 2 samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this fureturns an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.		
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterleaved.		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)	
writeArray	uInt16 []	The array of 16-bit samples to write to the task.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully written to the buffe		

#### Name Type Description

# DAQmxWriteCtrFreq

Writes a new pulse frequency and duty cycle to each channel in a continuous counter output task that contains one or more channels.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a valumore in order for the sample to write. If you pass a negative number, 1 returns an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if y start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the time lapses. A value of 0 indicates to try once to write the submitted samples. If the successfully writes all submitted samples, it does not return an error. (If the function returns a timeout error and returns the number of samples written.		
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterlea		
		Value	Descript	
		DAQmx_Val_GroupByChannel	Group by channel (nonintei	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleav	
frequency	float64 []	Specifies at what frequency to generate pulses.		
dutyCycle	float64 []	The width of the pulse divided by the pulse period. NI-DAQmx uses t combined with frequency, to determine both pulse width and pulse de		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
numSampsPerChanWritten	int32 *	The actual number of samples per channel successfully written to the		

#### Name Type Description

# DAQmxWriteCtrFreqScalar

int32 DAQmxWriteCtrFreqScalar (TaskHandle taskHandle, bool32 autoStart, float64 timeout, float64 frequency, float64 dutyCycle, bool32 \*reserved);

Writes a new pulse frequency and duty cycle to a continuous counter output task that contains a single channel.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to write the sample to.
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do not start it.
timeout	float64	The amount of time, in seconds, to wait for this function to write all the samples. The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this function successfully writes all submitted samples, it does not return an error. Otherwise, the function returns a timeout error and returns the number of samples actually written.
frequency	float64	Specifies at what frequency to generate pulses.
dutyCycle	float64	The width of the pulse divided by the pulse period. NI-DAQmx uses this ratio, combined with frequency, to determine both pulse width and pulse delay.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.

#### Name Type Description
# DAQmxWriteCtrTicks

Writes new pulse high tick counts and low tick counts to each channel in a continuous counter output task that contains one or more channels.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a valumore in order for the sample to write. If you pass a negative number, t returns an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if y start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the time lapses. A value of 0 indicates to try once to write the submitted samples. If the successfully writes all submitted samples, it does not return an error. (In the function returns a timeout error and returns the number of samples written.		
dataLayout	bool32	Specifies how the samples are arranged, either $\underline{interleaved}$	or noninterlea	
		Value	Descript	
		DAQmx_Val_GroupByChannel	Group by channel (nonintei	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleav	
highTicks	uInt32 []	The number of timebase ticks the pulse is high.		
lowTicks	uInt32 []	The number of timebase ticks the pulse is low.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
numSampsPerChanWritten	int32 *	The actual number of samples per channel successfully	written to the	

#### Name Type Description

# DAQmxWriteCtrTicksScalar

int32 DAQmxWriteCtrTicksScalar (TaskHandle taskHandle, bool32 autoStart, float64 timeout, uInt32 highTicks, uInt32 lowTicks, bool32 \*reserved);

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

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to write the sample to.
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do not start it.
timeout	float64	The amount of time, in seconds, to wait for this function to write all the samples. The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this function successfully writes all submitted samples, it does not return an error. Otherwise, the function returns a timeout error and returns the number of samples actually written.
highTicks	uInt32	The number of timebase ticks the pulse is high.
lowTicks	uInt32	The number of timebase ticks the pulse is low.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.

#### Name Type Description

## DAQmxWriteCtrTime

Writes a new pulse high time and low time to each channel in a continuous counter output task that contains one or more channels.

Input

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a valu more in order for the sample to write. If you pass a negative number, 1 returns an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if y start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the time lapses. A value of 0 indicates to try once to write the submitted samples. If the successfully writes all submitted samples, it does not return an error. (If the function returns a timeout error and returns the number of samples written.		
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterlea		
		Value	Descript	
		DAQmx_Val_GroupByChannel	Group by channel (nonintei	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleav	
highTime	float64 []	The amount of time the pulse is high.		
lowTime	float64 []	The amount of time the pulse is low.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
numSampsPerChanWritten	int32 *	The actual number of samples per channel successfully	written to the	

#### Name Type Description

# DAQmxWriteCtrTimeScalar

int32 DAQmxWriteCtrTimeScalar (TaskHandle taskHandle, bool32 autoStart, float64 timeout, float64 highTime, float64 lowTime, bool32 \*reserved);

Writes a new pulse high time and low time to a continuous counter output task that contains a single channel.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to write the sample to.
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do not start it.
timeout	float64	The amount of time, in seconds, to wait for this function to write all the samples. The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this function successfully writes all submitted samples, it does not return an error. Otherwise, the function returns a timeout error and returns the number of samples actually written.
highTime	float64	The amount of time the pulse is high.
lowTime	float64	The amount of time the pulse is low.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.

#### Name Type Description

# DAQmxWriteDigitalLines

Writes multiple samples to each digital line in a task. When you create your write array, each sample per channel must contain the number of bytes returned by the DAQmx\_Read\_DigitalLines\_BytesPerChan property.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of 2 samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this furtures an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if you de start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.		
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterleaved		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)	
writeArray	uInt8 []	The samples to write to the task.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully	written to the buffe	

#### Name Type Description

# DAQmxWriteDigitalScalarU32

int32 DAQmxWriteDigitalScalarU32 (TaskHandle taskHandle, bool32 autoStart, float64 timeout, uInt32 value, bool32 \*reserved);

Writes a single 32-bit unsigned *integer sample* to a task that contains a single *digital output channel*. Use this format for devices with up to 32 lines per port.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task to write the sample to.
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do not start it.
timeout	float64	The amount of time, in seconds, to wait for this function to write the <b>value</b> . The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the value. If this function successfully writes the value, it does not return an error. Otherwise, the function returns a timeout error.
value	uInt32	A <u>32-bit integer sample</u> to write to the task.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.

#### Name Type Description

# DAQmxWriteDigitalU8

int32 DAQmxWriteDigitalU8 (TaskHandle taskHandle, int32
numSampsPerChan, bool32 autoStart, float64 timeout, bool32
dataLayout, uInt8 writeArray[], int32 \*sampsPerChanWritten, bool32
\*reserved);

Writes multiple 8-bit unsigned integer samples to a task that contains one or more digital output channels. Use this format for devices with up to 8 lines per port.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of two samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this furtures an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.		
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterleaved.		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)	
writeArray	uInt8 []	The array of 8-bit integer samples to write to the task.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully	written to the buffe	

#### Name Type Description

# DAQmxWriteDigitalU16

Writes multiple 16-bit unsigned integer samples to a task that contains one or more digital output channels. Use this format for devices with up to 16 lines per port.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of two samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description		
taskHandle	TaskHandle	The task to write samples to.		
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this further returns an error.		
autoStart	bool32	Specifies whether or not this function automatically starts the task if you de start it.		
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.		
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterleaved		
		Value	Description	
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>	
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)	
writeArray	uInt16 []	The array of 16-bit integer samples to write to the task.		
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.		
Output				
Name	Туре	Description		
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully	written to the buffe	

#### Name Type Description

# DAQmxWriteDigitalU32

Writes multiple 32-bit unsigned integer samples to a task that contains one or more digital output channels. Use this format for devices with up to 32 lines per port.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of two samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description	
taskHandle	TaskHandle	The task to write samples to.	
numSampsPerChan	int32	The number of samples, per channel, to write. You must pass in a value of more in order for the sample to write. If you pass a negative number, this fureturns an error.	
autoStart	bool32	Specifies whether or not this function automatically starts the task if you de start it.	
timeout	float64	The amount of time, in seconds, to wait for this function to write all the sar The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this fur successfully writes all submitted samples, it does not return an error. Other the function returns a timeout error and returns the number of samples actu written.	
dataLayout	bool32	Specifies how the samples are arranged, either <i>interleaved</i> or noninterleaved.	
		Value	Description
		DAQmx_Val_GroupByChannel	Group by channel (noninterlea <sup>,</sup>
		DAQmx_Val_GroupByScanNumber	Group by sample (interleaved)
writeArray	uInt32 []	The array of 32-bit integer samples to write to the task.	
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.	
Output			
Name	Туре	Description	
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully written to the buffe	

#### Name Type Description
## DAQmxWriteRaw

Writes <sub>raw samples</sub> to a task. There is no scaling, parsing, or separation of the samples. Refer to the specifications for your instrument to determine the format of the incoming samples.



**Note** If you configured timing for your task, your write is considered a buffered write. Buffered writes require a minimum buffer size of 2 samples. If you do not configure the buffer size using DAQmxCfgOutputBuffer, NI-DAQmx automatically configures the buffer when you configure sample timing. If you attempt to write one sample for a buffered write without configuring the buffer, you will receive an error.

Name	Туре	Description
taskHandle	TaskHandle	The task to write samples to.
numSamps	int32	The number of samples per channel to write. You must pass in a value of 0 or more in order for the sample to write. If you pass a negative number, you will receive an error.
autoStart	bool32	Specifies whether or not this function automatically starts the task if you do not start it.
timeout	float64	The amount of time, in seconds, to wait for this function to write all the samples. The default value is 10.0 seconds. To specify an infinite wait, pass -1 (DAQmx_Val_WaitInfinitely). This function returns an error if the timeout elapses. A value of 0 indicates to try once to write the submitted samples. If this function successfully writes all submitted samples, it does not return an error. Otherwise, the function returns a timeout error and returns the number of samples actually written.
writeArray	void *	The raw samples to write to the task.
reserved	bool32 *	Reserved for future use. Pass NULL to this parameter.
Output		
Name	Туре	Description
sampsPerChanWritten	int32 *	The actual number of samples per channel successfully written to the buffer.

#### Name Type Description

# DAQmxExportSignal

int32 DAQmxExportSignal (TaskHandle taskHandle, int32 signalID, const char outputTerminal[]);

Routes a <u>control signal</u> to the specified <u>terminal</u>. The output terminal can reside on the device that generates the control signal or on a different device. Use this function to share clocks and triggers between multiple tasks and devices. The routes created by this function are <u>task-based routes</u>.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
signalID	int32	The name of the trigger, clock, or event to export.	
		Value	Description
		DAQmx_Val_AIConvertClock	Clock that causes an analog-to- digital conversion on an E Series or M Series device. One conversion corresponds to a single sample from one channel.
		DAQmx_Val_10MHzRefClock	Output of an oscillator that you can use to synchronize multiple devices.
		DAQmx_Val_20MHzTimebaseClock	Output of an oscillator that is the onboard source of the Master Timebase. Other timebases

	are derived from this clock.
DAQmx_Val_SampleClock	Clock the device uses to time each sample.
DAQmx_Val_AdvanceTrigger	Trigger that moves a switch to the next entry in a scan list.
DAQmx_Val_ReferenceTrigger	Trigger that establishes the reference point between pretrigger and posttrigger samples.
DAQmx_Val_StartTrigger	Trigger that begins a measurement or generation.
DAQmx_Val_AdvCmpltEvent	Signal that a switch product generates after it both executes the command(s) in a scan list entry and waits for the settling time

	to elapse.
DAQmx_Val_AIHoldCmpltEvent	Signal that an E Series or M Series device generates when the device latches analog input data (the ADC enters "hold" mode) and it is safe for any external switching hardware to remove the signal and replace it with the next signal. This event does not indicate the completion of the actual analog-to- digital
DAOmy Val CounterOutputEvent	Conversion.
DAQIIIA_vai_CounterOutputEvent	counter generates. Each time the counter reaches terminal

	count, this signal toggles or pulses.
DAQmx_Val_ChangeDetectionEvent	Signal that a static DIO device generates when the device detects a rising or falling edge on any of the lines or ports you selected when you configured change detection timing.
DAQmx_Val_WDTExpiredEvent	Signal that a static DIO device generates when the watchdog timer expires.

**outputTerminal** const char [] The destination terminal of the exported signal.

#### Name Type Description

# DAQmxCalculateReversePolyCoeff

int32 DAQmxCalculateReversePolyCoeff (float64 forwardCoeffs[], uInt32 numForwardCoeffsIn, float64 minValX, float64 maxValX, int32 numPointsToCompute, int32 reversePolyOrder, float64 reverseCoeffs[]);

Computes a set of coefficients for a polynomial that approximates the inverse of the polynomial with the coefficients you specify with **forwardCoeffs**. The function generates a table of *x* versus *y* values over the range of *x*. The function then finds a polynomial fit, using the least squares method to compute a polynomial that computes an *x* given a *y*.

Name	Туре	Description
forwardCoeffs	float64 []	The array of the forward coefficients for the polynomial that computes $y$ given a value of $x$ . Each element of the array corresponds to a term of the equation. For example, if index 3 of the array is 9, the fourth term of the equation is $9x^3$ .
numForwardCoeffsIn	uInt32	The number of elements in <b>forwardCoeff</b> .
minValX	float64	The minimum value of $x$ for which you use the polynomial. This is the smallest value of $x$ for which the function generates a $y$ value in the table.
maxValX	float64	The maximum value of $x$ for which you use the polynomial. This is the largest value of $x$ for which the function generates a $y$ value in the table.
numPointsToCompute	int32	The number of points in the table of <i>x</i> versus <i>y</i> values. The function spaces the values evenly between <b>minValX</b> and <b>maxValX</b> .
reversePolyOrder	int32	The order of the reverse polynomial to compute. For example, an input of 3 indicates a 3rd order polynomial. An input of -1 indicates to compute a reverse polynomial of the same order as the forward polynomial.
Output		
reverseCoeffs	float64 []	The array of the reverse coefficients of the polynomial. Each element of the array corresponds to a term of the equation. For example, if index 3 of the array is 9, the fourth term of the equation is 9y <sup>3</sup> . <b>reverseCoeffs</b> will be one greater than what you passed for <b>reversePolyOrder</b> . If you pass -1 for <b>reversePolyOrder</b> , the array must be the size of <b>forwardCoeffs</b> . Otherwise, the behavior is undefined.

#### Name Type Description

# DAQmxCreateLinScale

int32 DAQmxCreateLinScale (const char name[], float64 slope, float64 yIntercept, int32 preScaledUnits, const char scaledUnits[]);

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

Input

Name	Туре	Description
name	const char []	Identifies the custom scale for later use, such as when you create channels.
slope	float64	The slope, <i>m</i> , in the equation.
yIntercept	float64	The y-intercept, <i>b</i> , in the equation.
preScaledUnits	int32	The units of the values to scale.

Value	Description
DAQmx_Val_Volts	volts
DAQmx_Val_Amps	amperes
DAQmx_Val_DegF	degrees Fahrenheit
DAQmx_Val_DegC	degrees Celsius
DAQmx_Val_DegR	degrees Rankine
DAQmx_Val_Kelvins	kelvins
DAQmx_Val_Strain	strain
DAQmx_Val_Ohms	ohms
DAQmx_Val_Hz	hertz
DAQmx_Val_Seconds	seconds
DAQmx_Val_Meters	meters
DAQmx_Val_Inches	inches
DAQmx_Val_Degrees	degrees
DAQmx_Val_Radians	radians
DAQmx_Val_g	G. 1 g is approximately equal to 9.81 m/s <sup>2</sup> .
DAQmx_Val_Pascals	pascals
DAQmx_Val_FromTEDS	Units defined by TEDS information associated with the channel.

scaledUnits const char [] The units you want to use for the scaled value. You can use an arbitrary value.

#### Name Type Description

# DAQmxCreateMapScale

int32 DAQmxCreateMapScale (const char name[], float64 prescaledMin, float64
prescaledMax, float64 scaledMin, float64 scaledMax, int32
preScaledUnits, const char scaledUnits[]);

Creates and configures a scale that scales values proportionally from a range of prescaled values to a range of scaled values.

Input				
Name	Туре	Description		
name	const char []	Identifies the custom scale for later use, such as when you create channels.		
prescaledMin	float64	The smallest value in the range of prescaled values. NI-DAQmx maps this value to <b>scaledMin</b> .		
prescaledMax	float64	The largest value in the range of prescaled values. NI-DAQmx maps this value to <b>scaledMax</b> .		
scaledMin	float64	The smallest value in the range of scaled values. NI-DAQmx maps this value to <b>prescaledMin</b> . Read operations clip samples that are smaller than this value. Write operations generate errors for samples that are smaller than this value.		
scaledMax	float64	The largest value in the range of scaled values. NI-DAQmx maps this value to <b>prescaledMax</b> . Read operations clip samples that are larger than this value. Write operations generate errors for samples that are larger than this value.		
preScaledUnits	int32	The units of the values to scale.		
		Value	Description	
		DAQmx_Val_Volts	volts	
		DAQmx_Val_Amps	amperes	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegR	degrees Rankine	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_Strain	strain	
		DAQmx_Val_Ohms	ohms	
		DAQmx_Val_Hz	hertz	

DAQmx\_Val\_Pascals

DAQmx\_Val\_Seconds

DAQmx\_Val\_Meters

DAQmx\_Val\_Inches

DAQmx\_Val\_Degrees

DAQmx\_Val\_Radians

DAQmx\_Val\_g

pascals

seconds

meters

inches

degrees

radians

G. 1 g is approximately

equal to 9.81 m/s<sup>2</sup>.

## DAQmx\_Val\_FromTEDS

# Units defined by TEDS information associated with the channel.

scaledUnits const char [] The units you want to use for the scaled value. You can use an arbitrary value.

#### Name Type Description

# DAQmxCreatePolynomialScale

int32 DAQmxCreatePolynomialScale (const char name[], float64
forwardCoeffs[], uInt32 numForwardCoeffsIn, float64
reverseCoeffs[], uInt32 numReverseCoeffsIn, int32 preScaledUnits,
const char scaledUnits[]);

Creates and configures a scale that uses an *n*th order polynomial equation. NI-DAQmx requires both a polynomial to convert prescaled values to scaled values (forward) and a polynomial to convert scaled values to prescaled values (reverse). If you know only one set of coefficients, use <u>DAQmxCalculateReversePolyCoeff</u> to generate the other set.

Name	Туре	Description		
name	const char []	Identifies the custom scale for later use, such as when you create channels.		
forwardCoeffs	float64 []	An array of coefficients for the polynomial that converts prescaled values to scaled values. Each element of the array corresponds to a term of the equation. For example, if index 3 of the array is 9, the fourth term of the equation is $9x^3$ .		
numForwardCoeffsIn	uInt32	The number of elements in <b>forwardCoeff</b> .		
reverseCoeffs	float64 []	An array of coefficients for the polynomial that converts scaled values to prescaled values. Each element of the array corresponds to a term of the equation. For example, if index 3 of the array is 9, the fourth term of the equation is 9y <sup>3</sup> .		
numReverseCoeffsIn	uInt32	The number of elements in <b>reverseCoeff</b> .		
preScaledUnits	int32	The units of the values to scale.		
		Value	Description	
		DAQmx_Val_Volts	volts	
		DAQmx_Val_Amps	amperes	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegR	degrees Rankine	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_Strain	strain	
		DAQmx_Val_Ohms	ohms	
		DAQmx_Val_Hz	hertz	
		DAQmx_Val_Seconds	seconds	
		DAQmx_Val_Meters	meters	
		DAQmx_Val_Inches	inches	
		DAQmx_Val_Degrees	degrees	
		DAQmx_Val_Radians	radians	
		DAQmx_Val_g	G. 1 g is approximately	

	equal to 9.81 m/s <sup>2</sup> .			
DAQmx_Val_Pascals	pascals			
DAQmx_Val_FromTEDS	Units defined by TEDS information associated with the channel.			
'he units you want to use for the scaled value. You can use an arbitrary				

scaledUnits

const char [] The units you want to use for the scaled value. You can use an arbitrary value.

#### Name Type Description

# DAQmxCreateTableScale

int32 DAQmxCreateTableScale (const char name[], float64 prescaledVals[], uInt32 numPrescaledValsIn, float64 scaledVals[], uInt32 numScaledValsIn, int32 preScaledUnits, const char scaledUnits[]);

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

Input

Name	Туре	Description		
name	const char []	Identifies the custom scale for later use, such as when you create channels.		
prescaledVals	float64 []	An array of prescaled values. These values map directly to the values in <b>scaledVals</b> .		
numPrescaledValsIn	uInt32	The number of elements in <b>prescaledVals</b> .		
scaledVals	float64 []	An array of scaled values. These values map directly to the values in <b>prescaledVals</b> .		
numScaledValsIn	uInt32	The number of elements in <b>scaledVals</b> .		
preScaledUnits	int32	The units of the values to scale.		
		Value	Description	
		DAQmx_Val_Volts	volts	
		DAQmx_Val_Amps	amperes	
		DAQmx_Val_DegF	degrees Fahrenheit	
		DAQmx_Val_DegC	degrees Celsius	
		DAQmx_Val_DegR	degrees Rankine	
		DAQmx_Val_Kelvins	kelvins	
		DAQmx_Val_Strain	strain	
		DAQmx_Val_Ohms	ohms	
		DAQmx_Val_Hz	hertz	
		DAQmx_Val_Seconds	seconds	
		DAQmx_Val_Meters	meters	
		DAQmx_Val_Inches	inches	
		DAQmx_Val_Degrees	degrees	
		DAQmx_Val_Radians	radians	
		DAQmx_Val_g	G. 1 g is approximately equal to 9.81 m/s <sup>2</sup> .	
		DAQmx_Val_Pascals	pascals	
		DAQmx_Val_FromTEDS	Units defined by	

TEDS information associated with the channel.

scaledUnits

const char [] The units you want to use for the scaled value. You can use an arbitrary value.

#### Name Type Description

# DAQmxCfgInputBuffer

int32 DAQmxCfgInputBuffer (TaskHandle taskHandle, uInt32 numSampsPerChan);

Overrides the *automatic input buffer allocation* that NI-DAQmx performs.
Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
numSampsPerChan	uInt32	The number of samples the buffer can hold for each channel in the task. Zero indicates no buffer should be allocated. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxCfgOutputBuffer

Overrides the automatic output buffer allocation that NI-DAQmx performs.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
numSampsPerChan	uInt32	The number of samples the buffer can hold for each channel in the task. Zero indicates no buffer should be allocated. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxWaitForNextSampleClock

Waits until the next pulse of the <u>Sample Clock</u> occurs. If an extra Sample Clock pulse occurs between calls to this function, the second call returns an error or warning and waits for the next Sample Clock pulse. Use the get/set/reset functions for the <u>RealTimeConvLateErrorsToWarnings</u> property to specify whether this function returns errors or warnings.

Use this function to ensure <u>I/O cycles</u> complete within Sample Clock periods. National Instruments recommends you use this function for certain <u>applica tions</u> only.

Use the <u>DAQmx Real-Time</u> properties to configure error reporting and waiting options.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
timeout	float64	The maximum amount of time, in seconds, to wait for the next Sample Clock pulse. If the time elapses, this VI returns an error. The default timeout is 10 seconds. If you set <b>timeout</b> to -1, this VI waits indefinitely.
Output		
Name	Туре	Description
isLate	int32	A value of 0 indicates the read or write function executed in real time. A value of 1 indicates the function did not execute in real time.

#### Name Type Description

**isLate** int32 Indicates if this function detected an extra Sample Clock pulse after the specified number of warmup iterations execute. If you are using the C API, use <u>DAQmxSetRealTimeNumOfWarmupIters</u> to specify the number of warmup iterations. If you are using the CVI API, use DAQmxSetRealTimeAttribute to specify the number of iterations. This output is always FALSE until that number of loop iterations execute.

# DAQmxWaitUntilTaskDone

int32 DAQmxWaitUntilTaskDone (TaskHandle taskHandle, float64 timeToWait);

Waits for the measurement or generation to complete. Use this function to <u>ensure that</u> <u>the specified operation is complete</u> before you stop the task.

Input

1		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
timeToWait	float64	The maximum amount of time, in seconds, to wait for the measurement or generation to complete. The function returns an error if the time elapses before the measurement or generation is complete. A value of -1 (DAQmx_Val_WaitInfinitely) means to wait indefinitely. If you set <b>timeToWait</b> to 0, the function checks once and returns an error if the measurement or generation is not done.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# Obsolete

This function is obsolete and now always returns zero. The information below no longer applies but is provided for reference.

# DAQmxIsReadOrWriteLate

bool32 DAQmxIsReadOrWriteLate (int32 errorCode);

Returns TRUE if a DAQmx Read function or DAQmx Write function did not execute in real time.

Input
Name Type Description

**errorCode** int32 An error code or warning returned by one of the NI-DAQmx Library Read or Write functions.

#### Name Type Description

**isLate** int32 A value of 0 indicates the read or write function executed in real time. A value of 1 indicates the function did not execute in real time.

# DAQmxSwitchCloseRelays

int32 DAQmxSwitchCloseRelays (const char relays[], bool32 waitForSettling);

Closes the specified relays. If you set **waitForSettling** to TRUE, this function waits only after closing all relays. If you want to wait for settling after closing each relay, use this function multiple times to close each relay separately.

When you operate relays directly, you circumvent the protection that channel usage types offer. Avoid using this function when you use the DAQmxSwitchConnect function or the DAQmxSwitchDisconnect function. This function does not pass the changes you make to those functions.

Input		
Name	Туре	Description
relays	const char []	A set of <u>relays</u> to close.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchConnect

int32 DAQmxSwitchConnect (const char switchChannel1[], const char switchChannel2[], bool32 waitForSettling);

Makes a connection between two switch channels. When using this function, specify only the two connection endpoints using **switchChannel1** and **switchChannel2**. The function then attempts to find an available path between the two channels.

Input		
Name	Туре	Description
switchChannel1	const char []	The first channel to connect.
switchChannel2	const char []	The second channel to connect.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchConnectMulti

Makes one or more connections specified by a <u>connection list</u>. You can specify only the two endpoints, or you can specify the explicit path between two endpoints. This function can make connections on multiple devices, but each connection must reside on a single device. In the event of an error, connecting stops at the point in the list where the error occurred. If you set **waitForSettling** to TRUE, this function returns only after making all connections. If you want to wait for settling between connections, use this function multiple times to make connections separately.

Input		
Name	Туре	Description
connectionList	const char []	A list of connections, using a $\underline{\mbox{special syntax}},$ to make between switch channels.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchCreateScanList

int32 DAQmxSwitchCreateScanList (const char scanList[], TaskHandle
 \*taskHandle);

Creates a new switch scanning task that uses the specified scan list and applies it to the specified task.

Input		
Name	Туре	Description
scanList	const char []	Uses a special $_{\underline{\mathrm{syntax}}}$ to specify the sequence of connections and disconnections for the task.
Output		
Name	Туре	Description
taskHandle	TaskHandle *	The task created by this function.

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchDisconnect

int32 DAQmxSwitchDisconnect (const char switchChannel1[], const char switchChannel2[], bool32 waitForSettling);
Disconnects two switch channels. When using this function, specify only the two connection endpoints **switchChannel1** and **switchChannel2**.

Input		
Name	Туре	Description
switchChannel1	const char []	The first channel to disconnect.
switchChannel2	const char []	The second channel to disconnect.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchDisconnectAll

int32 DAQmxSwitchDisconnectAll (const char deviceName[], bool32 waitForSettling);

Terminates all active connections on the device, which places the relays into the topology reset state. You can use this function to terminate connections made by DAQmxSwitchConnect, DAQmxSwitchOpenRelays, Or DAQmxSwitchCloseRelays. This function does not alter the settling time or channel usage settings. To reset those settings, use DAQmxResetDevice Or DAQmxSwitchSetTopologyAndReset.

If you set **waitForSettling** to TRUE, this function returns only after terminating all connections. If you want to wait for settling between disconnections, use the <u>DAQmxSwitchDisconnect</u> function multiple times to terminate connections separately.

Input		
Name	Туре	Description
deviceName	const char []	The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchDisconnectMulti

int32 DAQmxSwitchDisconnectMulti (const char connectionList[], bool32 waitForSettling);

Terminates one or more connections specified by a <u>connection list</u>. This function can terminate connections on multiple devices. In the event of an error, the function stops at the point in the list where the error occurred. If you set **waitForSettling** to TRUE, this function returns only after terminating all connections. If you want to wait for settling between disconnections, use this function multiple times to terminate connections separately.

Input		
Name	Туре	Description
connectionList	const char []	Uses a special $\underline{syntax}$ to specify the list of switch connections to terminate.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchFindPath

int32 DAQmxSwitchFindPath (const char switchChannel1[], const char switchChannel2[], char path[], uInt32 pathBufferSize, int32 \*pathStatus);

Returns information about the path between **switchChannel1** and **switchChannel2**. If the channels are connected, this function returns the path by which they are connected. If the channels are not connected, the function returns a path by which they could be connected, if one is available.

Input		
Name	Туре	Description
switchChannel1	const char []	The first channel to connect.
switchChannel2	const char []	The second channel to connect.
pathBufferSize	uInt32	The size of $\ensuremath{\textbf{path}}$ . If you pass 0, this function returns the size of the buffer needed $t$
Output		
Name	Туре	Description
path	char []	The existing path or an available path between <b>switchChannel1</b> and <b>switchChan</b> the same <u>syntax</u> as a connection or disconnection list. If you pass NULL, this functi of the buffer needed to allocate.
pathStatus	int32 *	The status of the requested path. The following values can be returned:
		Value
		DAQmx_Val_PathStatus_Available

DAQmx\_Val\_PathStatus\_AlreadyExists

DAQmx\_Val\_PathStatus\_Unsupported

DAQmx\_Val\_PathStatus\_ChannelInUse

DAQmx\_Val\_PathStatus\_SourceChannelConflict

DAQmx\_Val\_PathStatus\_ChannelReservedForRouting

#### Name Type Description

status int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error. For this function, if you pass NULL for the buffer or 0 for the buffer size, this function returns the number of bytes needed to allocate.

# DAQmxSwitchGetMultiRelayCount

Returns the number of times a set of relays have been actuated. Use this function to track relay lifetime and usage. You must set the **waitForSettling** parameter of Switch functions to TRUE to obtain an accurate relay count.

Input		
Name	Туре	Description
relayList	const char []	The set of relays you want to query.
countArraySize	uInt32	The size of <b>count</b> . If you pass 0, this function returns the number of samples needed to allocate.
Output		
Name	Туре	Description
count	uInt32 []	Number of times each specified relay has been actuated. The order of this array depends on the order of <b>relayList</b> . If you pass NULL, this function returns the number of samples needed to allocate.
numRelayCountsRead	uInt32 *	The number of relay counts read by the function.

#### Name Type Description

status int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error. For this function, if you pass NULL for the buffer or 0 for the buffer size, this function returns the number of samples needed to allocate. For example, malloc (n\*sizeof(uInt32));

# DAQmxSwitchGetMultiRelayPos

int32 DAQmxSwitchGetMultiRelayPos (const char relayList[], uInt32 relayPos[], uInt32 relayPosArraySize, uInt32 \*numRelayPosRead);

Returns the current position of a set of relays.

Input			
Name	Туре	Description	
relayList	const char []	The set of relays you want to query.	
relayPosArraySize	uInt32	The size of <b>relayPos</b> . If you pass 0, this function returns the number of samples needed to allocate.	
Output			
Name	Туре	Description	
relayPos	uInt32 []	The position of each specified relay. The order of this array corresponds to the order of <b>relayList</b> . If you pass NULL, this function returns the number of samples needed to allocate.	
		Value	Description
		DAQmx_Val_Open	Relay is open.
		DAQmx_Val_Closed	Relay is closed.
numRelayPosRead uInt32 *		The number of relay positions read by	the function.

#### Name Type Description

status int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error. For this function, if you pass NULL for the buffer or 0 for the buffer size, this function returns the number of samples needed to allocate. For example, malloc (n\*sizeof(uInt32));

# DAQmxSwitchGetSingleRelayCount

Returns the number of times a single relay has been actuated. Use this function to track relay lifetime and usage. You must set the **waitForSettling** parameter of Switch functions to TRUE to obtain an accurate relay count.

Input		
Name	Туре	Description
relayName	const char []	The relay you want to query.
Output		
Name	Туре	Description
count	uInt32 *	The number of times the relay has been actuated.

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchGetSingleRelayPos

Returns the current position of a single relay.

Input			
Name	Туре	Description	
relayName	const char []	The relay you want to query.	
Output			
Name	Туре	Description	
relayPos	uInt32 *	The position of the relay.	
		Value	Description
		DAQmx_Val_Open	Relay is open.
		DAQmx_Val_Closed	Relay is closed.

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxSwitchOpenRelays

int32 DAQmxSwitchOpenRelays (const char relays[], bool32 waitForSettling);

Opens the specified relays. If you set **waitForSettling** to TRUE, this function waits only after opening all relays. If you want to wait for settling after opening each relay, use this function multiple times to open each relay separately.

When you operate relays directly, you circumvent the protection offered channel <u>usage types</u>. Avoid using this function when you use <u>DAQmxSwitchConnect</u> and <u>DAQmxSwitchDisconnect</u>. This function does not pass the changes you make to those functions.

Input		
Name	Туре	Description
relays	const char []	A set of <u>relays</u> to open.
waitForSettling	bool32	If TRUE, this function waits for the switches to settle before returning. If FALSE, the function returns immediately after the operation.

### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.
## DAQmxSwitchSetTopologyAndReset

## Purpose

Resets a switch device and sets its topology to the one specified with **newTopology**. Use this function to initialize a switch device before scanning or making connections to ensure the initial state of that switch.

This function differs from <u>DAQmxResetDevice</u> because it uses the input topology name rather than the one you specify in MAX. This function does not modify the channel usage defaults you configure in MAX.

### **Parameters**

Input

NameTypeDescriptiondeviceNameconst char[]The name of the device, as configured in Measurement & Automation Explorer (MAnewTopologyconst char[]The switch topology to use on the device. The following table lists examples.

# Value DAQmx Val Switch Topology 1127 1 Wire 64x1 Muz DAQmx Val Switch Topology 1127 2 Wire 32x1 Muz DAQmx\_Val\_Switch\_Topology\_1127\_2\_Wire\_4x8\_Matri DAQmx Val Switch Topology 1127 4 Wire 16x1 Muz DAQmx Val Switch Topology 1127 Independent DAQmx\_Val\_Switch\_Topology\_1128\_1\_Wire\_64x1\_Mux DAQmx\_Val\_Switch\_Topology\_1128\_2\_Wire\_32x1\_Mux DAQmx\_Val\_Switch\_Topology\_1128\_2\_Wire\_4x8\_Matri DAQmx\_Val\_Switch\_Topology\_1128\_4\_Wire\_16x1\_Mux DAQmx\_Val\_Switch\_Topology\_1128\_Independent DAQmx Val Switch Topology 1129 2 Wire 16x16 Ma DAQmx\_Val\_Switch\_Topology\_1129\_2\_Wire\_8x32\_Mat DAQmx Val Switch Topology 1129 2 Wire 4x64 Mat

DAQmx\_Val\_Switch\_Topology\_1129\_2\_Wire\_Dual\_8x10

DAQmx\_Val\_Switch\_Topology\_1129\_2\_Wire\_Dual\_4x3<sup>2</sup> DAQmx\_Val\_Switch\_Topology\_1129\_2\_Wire\_Quad\_4x1

DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_256x1\_Mi DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_Dual\_128: DAQmx\_Val\_Switch\_Topology\_1130\_2\_Wire\_128x1\_Mi DAQmx\_Val\_Switch\_Topology\_1130\_4\_Wire\_64x1\_Mu: DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_4x64\_Mat DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_8x32\_Mat DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_Octal\_32x DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_Quad\_64x DAQmx\_Val\_Switch\_Topology\_1130\_1\_Wire\_Quad\_64x

DAQmx\_Val\_Switch\_Topology\_1130\_2\_Wire\_4x32\_Mat DAQmx\_Val\_Switch\_Topology\_1130\_2\_Wire\_Octal\_16x DAQmx\_Val\_Switch\_Topology\_1130\_2\_Wire\_Quad\_32x DAQmx\_Val\_Switch\_Topology\_1130\_4\_Wire\_Quad\_16x DAQmx\_Val\_Switch\_Topology\_1130\_Independent DAQmx\_Val\_Switch\_Topology\_1160\_16\_SPDT DAQmx\_Val\_Switch\_Topology\_1161\_8\_SPDT DAQmx\_Val\_Switch\_Topology\_1163R\_Octal\_4x1\_Mux

DAQmx\_Val\_Switch\_Topology\_1166\_32\_SPDT DAQmx\_Val\_Switch\_Topology\_1167\_Independent DAQmx\_Val\_Switch\_Topology\_1169\_100\_SPST DAQmx\_Val\_Switch\_Topology\_1175\_1\_Wire\_196x1\_Mt

DAQmx\_Val\_Switch\_Topology\_1175\_2\_Wire\_98x1\_Muz

DAQmx\_Val\_Switch\_Topology\_1175\_2\_Wire\_95x1\_Muz

DAQmx\_Val\_Switch\_Topology\_1190\_Quad\_4x1\_Mux

DAQmx\_Val\_Switch\_Topology\_1191\_Quad\_4x1\_Mux

DAQmx\_Val\_Switch\_Topology\_1192\_8\_SPDT DAQmx\_Val\_Switch\_Topology\_1193\_32x1\_Mux DAQmx\_Val\_Switch\_Topology\_1193\_Dual\_16x1\_Mux

DAQmx\_Val\_Switch\_Topology\_1193\_Quad\_8x1\_Mux

DAQmx\_Val\_Switch\_Topology\_1193\_16x1\_Terminated\_

DAQmx\_Val\_Switch\_Topology\_1193\_Dual\_8x1\_Termina

DAQmx\_Val\_Switch\_Topology\_1193\_Quad\_4x1\_Termin

DAQmx\_Val\_Switch\_Topology\_1193\_Independent DAQmx\_Val\_Switch\_Topology\_1194\_Quad\_4x1\_Mux DAQmx\_Val\_Switch\_Topology\_1195\_Quad\_4x1\_Mux DAQmx\_Val\_Switch\_Topology\_2501\_1\_Wire\_48x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2501\_1\_Wire\_48x1\_Am

DAQmx\_Val\_Switch\_Topology\_2501\_2\_Wire\_24x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2501\_2\_Wire\_24x1\_Am

DAQmx Val Switch Topology 2501 2 Wire Dual 12x DAQmx Val Switch Topology 2501 2 Wire Quad 6x1 DAQmx Val Switch Topology 2501 2 Wire 4x6 Matr DAQmx\_Val\_Switch\_Topology\_2501\_4\_Wire\_12x1 Mu: DAQmx\_Val\_Switch\_Topology\_2503\_1\_Wire\_48x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2503\_2\_Wire\_24x1\_Mu: DAQmx Val Switch Topology 2503 2 Wire Dual 12x DAQmx\_Val\_Switch\_Topology\_2503\_2\_Wire\_Quad\_6x1 DAQmx\_Val\_Switch\_Topology\_2503\_2\_Wire\_4x6\_Matr DAQmx\_Val\_Switch\_Topology\_2503\_4\_Wire\_12x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2527\_1\_Wire\_64x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2527\_1\_Wire\_Dual\_32x DAQmx\_Val\_Switch\_Topology\_2527\_2\_Wire\_32x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2527\_2\_Wire\_Dual\_16x DAQmx Val Switch Topology 2527 4 Wire 16x1 Mu: DAQmx Val Switch Topology 2527 Independent DAQmx\_Val\_Switch\_Topology\_2529\_2\_Wire\_8x16\_Mat DAQmx\_Val\_Switch\_Topology\_2529\_2\_Wire\_4x32\_Mat DAQmx\_Val\_Switch\_Topology\_2529\_2\_Wire\_Dual\_4x10 DAQmx Val Switch Topology 2530 1 Wire 128x1 Mi DAQmx Val Switch Topology 2530 1 Wire Dual 64x DAQmx\_Val\_Switch\_Topology\_2530\_2\_Wire\_64x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2530\_4\_Wire\_32x1\_Mu: DAQmx\_Val\_Switch\_Topology\_2530\_1\_Wire\_4x32\_Mat DAQmx\_Val\_Switch\_Topology\_2530\_1\_Wire\_8x16\_Mat DAQmx\_Val\_Switch\_Topology\_2530\_1\_Wire\_Octal\_16x DAQmx\_Val\_Switch\_Topology\_2530\_1\_Wire\_Quad\_32x DAQmx\_Val\_Switch\_Topology\_2530\_2\_Wire\_4x16\_Mat DAQmx\_Val\_Switch\_Topology\_2530\_2\_Wire\_Dual\_32x DAQmx\_Val\_Switch\_Topology\_2530\_2\_Wire\_Quad\_16x DAQmx\_Val\_Switch\_Topology\_2530\_4\_Wire\_Dual\_16x DAQmx\_Val\_Switch\_Topology\_2530\_Independent DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_16x32\_M; DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_4x128\_M; DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_8x64\_Mat DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_8x64\_Mat

DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_Dual\_4x6

DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_Dual\_8x3

DAQmx\_Val\_Switch\_Topology\_2532\_1\_Wire\_Sixteen\_2

DAQmx\_Val\_Switch\_Topology\_2532\_2\_Wire\_16x16\_Ma

DAQmx\_Val\_Switch\_Topology\_2532\_2\_Wire\_4x64\_Mat

DAQmx\_Val\_Switch\_Topology\_2532\_2\_Wire\_8x32\_Mat

DAQmx\_Val\_Switch\_Topology\_2564\_16\_SPST DAQmx\_Val\_Switch\_Topology\_2565\_16\_SPST DAQmx\_Val\_Switch\_Topology\_2566\_16\_SPDT DAQmx\_Val\_Switch\_Topology\_2567\_Independent DAQmx\_Val\_Switch\_Topology\_2568\_31\_SPST DAQmx\_Val\_Switch\_Topology\_2569\_100\_SPST DAQmx\_Val\_Switch\_Topology\_2570\_40\_SPDT DAQmx\_Val\_Switch\_Topology\_2575\_1\_Wire\_196x1\_Mi

DAQmx\_Val\_Switch\_Topology\_2575\_2\_Wire\_98x1\_Mu:

DAQmx\_Val\_Switch\_Topology\_2575\_2\_Wire\_95x1\_Mu:

DAQmx\_Val\_Switch\_Topology\_2576\_2\_Wire\_64x1\_Mux

DAQmx\_Val\_Switch\_Topology\_2576\_2\_Wire\_Dual\_32x

DAQmx\_Val\_Switch\_Topology\_2576\_2\_Wire\_Octal\_8x1

DAQmx\_Val\_Switch\_Topology\_2576\_2\_Wire\_Quad\_16x

DAQmx\_Val\_Switch\_Topology\_2576\_2\_Wire\_Sixteen\_4

DAQmx\_Val\_Switch\_Topology\_2576\_Independent DAQmx\_Val\_Switch\_Topology\_2585\_1\_Wire\_10x1\_Mux

DAQmx\_Val\_Switch\_Topology\_2586\_10\_SPST DAQmx\_Val\_Switch\_Topology\_2590\_4x1\_Mux DAQmx\_Val\_Switch\_Topology\_2591\_4x1\_Mux DAQmx\_Val\_Switch\_Topology\_2593\_16x1\_Mux DAQmx\_Val\_Switch\_Topology\_2593\_Dual\_8x1\_Mux

DAQmx\_Val\_Switch\_Topology\_2593\_8x1\_Terminated\_N

DAQmx\_Val\_Switch\_Topology\_2593\_Dual\_4x1\_Termina

DAQmx\_Val\_Switch\_Topology\_2593\_Independent DAQmx\_Val\_Switch\_Topology\_2594\_4x1\_Mux DAQmx\_Val\_Switch\_Topology\_2595\_4x1\_Mux DAQmx\_Val\_Switch\_Topology\_2596\_Dual\_6x1\_Mux

DAQmx\_Val\_Switch\_Topology\_2597\_6x1\_Terminated\_N

DAQmx\_Val\_Switch\_Topology\_2598\_Dual\_Transfer

DAQmx\_Val\_Switch\_Topology\_2599\_2\_SPDT

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxSwitchWaitForSettling

int32 DAQmxSwitchWaitForSettling (const char deviceName[]);

## Purpose

Waits for the settling time on the device to expire. The device resets this time and begins counting down again when the device performs an operation. This function can return immediately if no operation happened recently.

### **Parameters**

Input

Name Type Description

**deviceName** const char [] The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxConnectTerms

int32 DAQmxConnectTerms (const char sourceTerminal[], const char destinationTerminal[], int32 signalModifiers);

## Purpose

Creates a route between a source and destination <u>terminal</u>. The route can carry a variety of digital signals, such as triggers, clocks, and hardware events.

These source and destination terminals can be on different devices as long as a connecting public bus, such as RTSI or the PXI backplane, is available. DAQmxConnectTerms does not modify a task. When DAQmxConnectTerms runs, the route is immediately reserved and committed to hardware. This type of routing is called <u>immediate routing</u>.

## **Parameters**

Input				
Name	Туре	Description		
sourceTerminal	const char []	The originating <i>terminal</i> of the route. You can specify a <i>terminal name</i> .		
destinationTerminal	const char []	The receiving <i>terminal</i> of the route. You can specify a <i>terminal name</i> .		
signalModifiers	int32	Specifies whether or not to invert the signal routed from the <b>sourceTerminal</b> to the <b>destinationTerminal</b> . If the device is not capable of signal inversion or if a previous route reserved the inversion circuitry in an incompatible configuration, attempting to invert the signal causes an error.		
		Value	Description	
		DAQmx_Val_DoNotInvertPolarity	Do not invert the signal.	
		DAQmx_Val_InvertPolarity	Invert the signal.	

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxDisconnectTerms

int32 DAQmxDisconnectTerms (const char sourceTerminal[], const char destinationTerminal[]);

## Purpose

Removes signal routes previously created using <u>DAQmxConnectTerms</u>.

DAQmxDisconnectTerms cannot remove task-based routes, such as those created through timing and triggering configuration.

When this function executes, the route is unreserved immediately. For this reason, this type of routing is called <u>immediate routing</u>.

## **Parameters**

Input		
Name	Туре	Description
sourceTerminal	const char []	The originating $\underline{\operatorname{terminal}}$ of the route. You can specify a $\underline{\operatorname{terminal}}$ name.
destinationTerminal	const char []	The receiving <i>terminal</i> of the route. You can specify a <i>terminal name</i> .

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxTristateOutputTerm

int32 DAQmxTristateOutputTerm (const char outputTerminal[]);

## Purpose

Sets a terminal to <u>high-impedance state</u>. If you connect an external signal to a terminal on the I/O connector, the terminal must be in high-impedance state. Otherwise, the device could double-drive the terminal and damage the hardware. If you use this function on a terminal in an active route, the function fails and returns an error.

<u>DAQmxResetDevice</u> sets all terminals on the I/O connector to high-impedance state but aborts any running tasks associated with the device.

## **Parameters**

Input Name

Туре

Description

**outputTerminal** const char [] The terminal on the I/O connector to set to high-impedance state. You can specify a <u>terminal name</u>.

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxResetDevice

int32 DAQmxResetDevice (const char deviceName[]);

## Purpose

Immediately aborts all tasks associated with a device and returns the device to an initialized state. Aborting a task stops and releases any resources the task reserved.

### **Parameters**

Input

Name Type Description

**deviceName** const char [] The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxControlWatchdogTask

int32 DAQmxControlWatchdogTask (TaskHandle taskHandle, int32 action);

## Purpose

Controls the watchdog task according to the action you specify.

## Parameters

Input

Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
action	int32	Specifies how to control the watchdog task.	
		Value	Description
		DAQmx_Val_ResetTimer	Resets the internal timer to 0. You must continually reset the internal timer to prevent it from timing out and locking out the device.
		DAQmx_Val_ClearExpiration	Unlocks a device whose watchdog expired.

## **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

## DAQmxCreateWatchdogTimerTask

int32 DAQmxCreateWatchdogTimerTask (const char deviceName[], const char taskName[], TaskHandle \*taskHandle, float64 timeout, const char channelName[], int32 expState, ...);
Creates and configures a task that controls the watchdog timer of a device. The timer activates when you start the task.

Input

Name	Туре	Description	
deviceName	const char []	The name of the device, as configured in M Automation Explorer (MAX), to which this	leasurement & s operation applies.
taskName	const char []	The name to assign to the task. If you use the specify a name for the task, you must use $\Box$ loop after you are finished with the task. Or attempts to create multiple tasks with the satisfient of the satisfien	his function in a loop and <u>AQmxClearTask</u> within the therwise, NI-DAQmx ame name, which results
timeout	float64	The time, in seconds, until the watchdog tin DAQmx_Val_WaitInfinitely indicates that expires. Use DAQmx_Val_WaitInfinitely w signal an expiration of the timer. If this tim the physical channels to the states you spec Use <u>DAQmxControlWatchdogTask</u> with <b>action</b> so DAQmx_Val_ResetTimer to prevent the water expiring.	mer expires. A value of the internal timer never when you use a trigger to e elapses, the device sets ify in <b>expState</b> . et to atchdog timer from
channelName	const char []	The digital line or port to modify. You can digital input lines. You can specify a <u>list or ra</u>	not modify dedicated ange of physical channels
expState	int32	The state to which to set the digital physica watchdog timer expires.	l channel when the
		Value	Description
		DAQmx_Val_High	High logic
		DAQmx_Val_Low	Low logic
		DAQmx_Val_Tristate	High- impedance state. You can select this state only on devices with bidirectional ports, and you can select it only for entire ports. You cannot select this state for dedicated digital output lines.

		DAQmx_Val_NoChange	Expiration does not affect the port. Do not change the state of any lines in the port, and do not lock the port. For example, if a line is high when the timer expires, that line stays high, and you can write new values to the line. You can select this value only for entire ports.
moreChannelsAndStates	any type (passed by value)	Pairs of additional channels and the states to the device powers up or when the device is NULL at the end of the argument list. If you do not want to pass additional channel function call can be similar to the following DAQmxCreateWatchdogTimerTask ("Dev1 *taskHandle, DAQmx_Val_WaitInfinitely, DAQmx_Val_High, NULL); If you pass additional channels and states, to similar to the following example: DAQmxCreateWatchdogTimerTask ("Dev1 *taskHandle, DAQmx_Val_WaitInfinitely, DAQmx_Val_High, "Dev1/do1", DAQmx_Val_High, "Dev1/do1", DAQmx_Val_Market	<pre>b set the channels to when reset. You must pass els and states, the g example: _", "myTask", "Dev1/do0", he function call can be L", "myTask", "Dev1/do0", _Val_Tristate, NULL);</pre>
Output			
taskHandle	TaskHandle *	A reference to the new task.	

### Name Type Description

# DAQmxAOSeriesCalAdjust

DAQmxAOSeriesCalAdjust (uInt32 calHandle, float64 referenceVoltage);

Adjusts the <u>external calibration</u> constants on an AO Series device. You must <u>supply a known voltage</u> <u>to the device</u> and specify that voltage with **referenceVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceVoltage	float64	The known voltage, in volts, to use as a reference for calibration. This voltage should be between +6.000 V and +9.999 V.

### Name Type Description

# DAQmxESeriesCalAdjust

int32 DAQmxESeriesCalAdjust (uInt32 calHandle, float64 referenceVoltage);

Adjusts the *external calibration* constants on an E-Series device. You must *supply a known voltage to the device* and specify that voltage with **referenceVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceVoltage	float64	The known voltage, in volts, to use as a reference for calibration. This voltage should be between +6.000 V and +9.999 V.

### Name Type Description

# DAQmxGetAIChanCalCalDate

int32 DAQmxGetAIChanCalCalDate (TaskHandle taskHandle, const char channelName[], uInt32 \*year, uInt32 \*month, uInt32 \*day, uInt32 \*hour, uInt32 \*minute);

Indicates the last date and time that the channel underwent a channel calibration.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
channelName	const char []	Name of the local or global channel to query calibration for.
Output		
Name	Туре	Description
year	uInt32 *	The last year that the channel underwent a channel calibration.
month	uInt32 *	The last month that the channel underwent a channel calibration.
day	uInt32 *	The last day that the channel underwent a channel calibration.
hour	uInt32 *	The last hour, on a 24-hour clock, that the channel underwent a channel calibration.
minute	uInt32 *	The last minute that the channel underwent a channel calibration.

### Name Type Description

# DAQmxGetAIChanCalExpDate

int32 DAQmxGetAIChanCalExpDate (TaskHandle taskHandle, const char channelName[], uInt32 \*year, uInt32 \*month, uInt32 \*day, uInt32 \*hour, uInt32 \*minute);

Indicates the last date and time that the channel underwent an external channel calibration.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
channelName	const char []	Name of the local or global channel to query calibration for.
Output		
Name	Туре	Description
year	uInt32 *	The last year that the channel underwent an external channel calibration.
month	uInt32 *	The last month that the channel underwent an external channel calibration.
day	uInt32 *	The last day that the channel underwent an external channel calibration.
hour	uInt32 *	The last hour, on a 24-hour clock, that the channel underwent an external channel calibration.
minute	uInt32 *	The last minute that the channel underwent an external channel calibration.

### Name Type Description

# DAQmxGetExtCalLastDateAndTime

int32 DAQmxGetExtCalLastDateAndTime (const char deviceName[], uInt32
\*year, uInt32 \*month, uInt32 \*day, uInt32 \*hour, uInt32 \*minute);

Indicates the last date and time that the device underwent an external calibration.

Input		
Name	Туре	Description
deviceName	const char []	The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.
Output		
Name	Туре	Description
year	uInt32 *	The last year that the device underwent an external calibration.
month	uInt32 *	The last month that the device underwent an external calibration.
day	uInt32 *	The last day that the device underwent an external calibration.
hour	uInt32 *	The last hour, on a 24-hour clock, that the device underwent an external calibration.
minute	uInt32 *	The last minute that the device underwent an external calibration.

### Name Type Description

# DAQmxGetSelfCalLastDateAndTime

int32 DAQmxGetSelfCalLastDateAndTime (const char deviceName[], uInt32
\*year, uInt32 \*month, uInt32 \*day, uInt32 \*hour, uInt32 \*minute);

Indicates the last date and time that the device underwent a self calibration.

Input		
Name	Туре	Description
deviceName	const char []	The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.
Output		
Name	Туре	Description
year	uInt32 *	The last year that the device underwent a self-calibration.
month	uInt32 *	The last month that the device underwent a self-calibration.
day	uInt32 *	The last day that the device underwent a self-calibration.
hour	uInt32 *	The last hour, on a 24-hour clock, that the device underwent a self-calibration.
minute	uInt32 *	The last minute that the device underwent a self-calibration.

### Name Type Description

# DAQmxMSeriesCalAdjust

int32 DAQmxMSeriesCalAdjust (uInt32 calHandle, float64 referenceVoltage);

Adjusts the *external calibration* constants for an M Series device. You must connect a known voltage to the device and specify that voltage with **referenceVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceVoltage	float64	The known voltage, in volts, to use as a reference for calibration. This voltage should be between +6.000 V and +9.999 V.

### Name Type Description

# DAQmxPerformBridgeOffsetNullingCal

Performs a bridge offset nulling calibration on the channels in the task. If the task measures both bridge-based sensors and non-bridge-based sensors, specify the names of the channels that measure bridge-based sensors in the **channel** parameter.

Input		
Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
channel	const char []	A subset of virtual channels in the task that you want to calibrate. Use this parameter if you do not want to calibrate all the channels in the task or if some channels in the task measure non-bridge-based sensors.

### Name Type Description
# DAQmxPerformBridgeOffsetNullingCalEx

int32 DAQmxPerformBridgeOffsetNullingCalEx (TaskHandle taskHandle, const char channel[], bool32 skipUnsupportedChannels)

Performs a bridge offset nulling calibration on the channels in the task. If the task measures both bridge-based sensors and non-bridge-based sensors, specify the names of the channels that measure bridge-based sensors in the **channel** parameter.

Input

Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
channel	const char []	A subset of virtual channels in the task that you want to calibrate. Use this parameter if you do not want to calibrate all the channels in the task or if some channels in the task measure non-bridge-based sensors.
skipUnsupportedChannels	bool32	Specifies whether or not to skip channels that do not support calibration. If <b>skipUnsupportedChannels</b> is TRUE, calibration will be performed only on supported channels. If FALSE, calibration will be performed on channels specified by <b>channel</b> . The default is FALSE.

#### Name Type Description

# DAQmxPerformBridgeShuntCal

int32 DAQmxPerformBridgeShuntCal (TaskHandle taskHandle, const char channel[], float64 shuntResistorValue, int32 shuntResistorLocation, float64 bridgeResistance, bool32 skipUnsupportedChannels);

Performs shunt calibration for the specified channels using a bridge sensor.

Input

Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
channel	const char []	A subset of virtual channels in the task that y Use this parameter if you do not want to cali the task or if some channels in the task meas sensors. If the input is empty, the function w shunt calibration on all the channels in the ta	you want to calibrate. ibrate all the channels in sure non-bridge-based rill attempt to perform ask.
shuntResistorValue	float64	The resistance, in ohms, of the shunt resistor	r.
shuntResistorLocation	int32	The location of the shunt resistor. Refer to the more information on <u>bridge configurations</u> .	ne NI-DAQmx Help for
		Value	Description
		DAQmx_Val_R1 (12465)	R1
		DAQmx_Val_R2 (12466)	R2
		DAQmx_Val_R3 (12467)	R3
		DAQmx_Val_R4 (14813)	R4
bridgeResistance	float64	The resistance, in ohms, of the bridge sensor	r.
skipUnsupportedChannels	bool32	Specifies whether or not to skip channels that calibration. If <b>skipUnsupportedChannels</b> is calibration will be performed only on support shunt calibration will be performed on channel <b>channelNames</b> . The default is FALSE.	at do not support shunt s TRUE, shunt rted channels. If FALSE, nels specified by

#### Name Type Description

# DAQmxPerformStrainShuntCal

int32 DAQmxPerformStrainShuntCal (TaskHandle taskHandle, const char channel[], float64 shuntResistorValue, int32 shuntResistorLocation, bool32 skipUnsupportedChannels);

Performs shunt calibration for the specified channels using a strain gage sensor.

Input

Name	Туре	Description	
taskHandle	TaskHandle	The task used in this function.	
channel	const char []	A subset of virtual channels in the task that y Use this parameter if you do not want to cal the task or if some channels in the task meas sensors. If the input is empty, the function w shunt calibration on all the channels in the task	you want to calibrate. ibrate all the channels in sure non-bridge-based vill attempt to perform ask.
shuntResistorValue	float64	The resistance, in ohms, of the shunt resisto	r.
shuntResistorLocation	int32	The location of the shunt resistor. Refer to the more information on <u>bridge configurations</u> .	he NI-DAQmx Help for
		Value	Description
		DAQmx_Val_R1 (12465)	R1
		DAQmx_Val_R2 (12466)	R2
		DAQmx_Val_R3 (12467)	R3
		DAQmx_Val_R4 (14813)	R4
skipUnsupportedChannels	bool32	Specifies whether or not to skip channels the calibration. If <b>skipUnsupportedChannels</b> is calibration will be performed only on support shunt calibration will be performed on channelNames. The default is FALSE.	at do not support shunt s TRUE, shunt rted channels. If FALSE, nels specified by

#### Name Type Description

# ${\bf DAQmx} Restore Last Ext CalConst$

int32 DAQmxRestoreLastExtCalConst (const char deviceName[]);

Sets the <u>self calibration constants</u> of the device to the the current <u>external calibration constants</u>. National Instruments sets the external calibration constants at the factory, and those constants remain in effect until you perform a new external calibration on the device.

This function nullifies any self calibration you perform on the device. If you have never performed a self calibration on the device, this function has no effect.

Input

Name Type Description

**deviceName** const char [] The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.

#### Name Type Description

# DAQmxSCBaseboardCalAdjust

int32 DAQmxSCBaseboardCalAdjust (uInt32 calHandle, float64 referenceVoltage);

Adjusts the *external calibration* constants on for the baseboard of an SC Series device. You must connect a known voltage to the device and specify that voltage with **referenceVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceVoltage	float64	The known voltage, in volts, to use as a reference for calibration. This voltage should be between +6.000 V and +9.999 V.

#### Name Type Description

# DAQmxSelfCal

int32 DAQmxSelfCal (const char deviceName[]);

Measures the onboard reference voltage of the device and adjusts the <u>self-calibration</u> constants to account for any errors caused by short-term fluctuations in the operating environment. When you self calibrate a device, no external signal connections are necessary.

Input

Name Type Description

**deviceName** const char [] The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.

#### Name Type Description

# DAQmxSetAIChanCalCalDate

int32 DAQmxSetAIChanCalCalDate (TaskHandle taskHandle, const char channelName[], uInt32 \*year, uInt32 \*month, uInt32 \*day, uInt32 \*hour, uInt32 \*minute);

Sets the date and time that the channel underwent a channel calibration.

Input

Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
channelName	const char []	Name of the local or global channel to calibrate.
year	uInt32	The last year that the channel underwent a channel calibration.
month	uInt32	The last month that the channel underwent a channel calibration.
day	uInt32	The last day that the channel underwent a channel calibration.
hour	uInt32	The last hour, on a 24-hour clock, that the channel underwent a channel calibration.
minute	uInt32	The last minute that the channel underwent a channel calibration.

#### Name Type Description

# DAQmxSetAIChanCalExpDate

int32 DAQmxSetAIChanCalExpDate (TaskHandle taskHandle, const char channelName[], uInt32 \*year, uInt32 \*month, uInt32 \*day, uInt32 \*hour, uInt32 \*minute);

Sets the date and time that the channel underwent an external channel calibration.

Input

Name	Туре	Description
taskHandle	TaskHandle	The task used in this function.
channelName	const char []	Name of the local or global channel to calibrate.
year	uInt32	The last year that the channel underwent an external channel calibration.
month	uInt32	The last month that the channel underwent an external channel calibration.
day	uInt32	The last day that the channel underwent an external channel calibration.
hour	uInt32	The last hour, on a 24-hour clock, that the channel underwent an external channel calibration.
minute	uInt32	The last minute that the channel underwent an external channel calibration.

#### Name Type Description

# DAQmxSSeriesCalAdjust

int32 DAQmxSSeriesCalAdjust (uInt32 calHandle, float64 referenceVoltage);

Adjusts the *external calibration* constants for an S Series device. You must connect a known voltage to the device and specify that voltage with **referenceVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceVoltage	float64	The known voltage, in volts, to use as a reference for calibration. This voltage should be between +6.000 V and +9.999 V.

#### Name Type Description
# DAQmxChangeExtCalPassword

int32 DAQmxChangeExtCalPassword (const char deviceName[], const char password[], const char newPassword[]);

Changes the external calibration password of the device.

Input

Name	Туре	Description
deviceName	const char []	The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.
password	const char []	The current calibration password for the device. This password is case sensitive. The default password for all NI products is NI.
newPassword	const char []	The new password for the device. This password can be no longer than four characters.

#### Name Type Description

# DAQmxCloseExtCal

int32 DAQmxCloseExtCal (uInt32 calHandle, int32 action);

Closes an open *external calibration* session.

Input

Name Type Description

action int32

calHandle uInt32 A reference to the calibration session that you created using the DAQmxInitExtCal function.

Specifies how to close the calibration session.

Value	Description
DAQmx_Val_Action_Commit	Saves the calibration changes made in the session
DAQmx_Val_Action_Cancel	Closes the session without saving any calibration changes

#### Name Type Description

# DAQmxInitExtCal

int32 DAQmxInitExtCal (const char deviceName[], const char password[], uInt32 \*calHandle);

Starts an *external calibration* session on a device.

Input		
Name	Туре	Description
deviceName	const char []	The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.
password	const char []	The current calibration password for the device. This password is case sensitive. The default password for all NI products is NI.
Output		
Name	Туре	Description
calHandle	uInt32 *	A reference to the calibration session.

#### Name Type Description

# DAQmxAdjustDSAAICal

int32 DAQmxAdjustDSAAICal (uInt32 calHandle, float64 referenceVoltage);

Adjusts the *external calibration* constants for the analog input section of a DSA device. You must connect a known voltage to the device and specify that voltage with the **referenceVoltage** parameter.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceVoltage	float64	The known voltage, in volts, to use as a reference for calibration. This voltage should be between +4.9 V and +9.1 V.

#### Name Type Description

# DAQmxAdjustDSAAOCal

int32 DAQmxAdjustDSAAOCal (uInt32 calHandle, uInt32 channel, float64 requestedLowVoltage, float64 actualLowVoltage, float64 requestedHighVoltage, float64 actualHighVoltage, float64 gainSetting);

Adjusts the external calibration constants for the analog output section of a DSA device. You must use the device to generate a high voltage and low voltage at a specified gain, measure the high and low voltages, then specify the requested high voltage, low voltage, and gain along with the actual high voltage and low voltage.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channel	uInt32	The number of the channel to calibrate. This number is the numeric portion of the physical channel name, not the full physical channel name.
requestedLowVoltage	float64	The low voltage you attempted to generate at the gain setting you specified.
actualLowVoltage	float64	The actual low voltage as measured by an external sensor.
requestedHighVoltage	float64	The high voltage you attempted to generate at the gain setting you specified.
actualHighVoltage	float64	The actual high voltage as measured by an external sensor.
gainSetting	float64	The gain setting you used when you attempted to generate the requested high voltage and requested low voltage.

#### Name Type Description

# DAQmxAdjustDSATimebaseCal

int32 DAQmxAdjustDSATimebaseCal (uInt32 calHandle, float64 referenceFrequency);

Adjusts the *external calibration* constant for the timebase of a DSA device with an adjustable oscillator. You must connect a sinusoidal signal with a known frequency to the device and specify that frequency with **referenceFrequency**.

Input

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
referenceFrequency	float64	The frequency, in hertz, of the signal to use as a reference for calibration.

#### Name Type Description

# DAQmxAdjust4204Cal

int32 DAQmxAdjust4204Cal (uInt32 calHandle, const char channelNames[], float64 lowPassFreq, bool32 trackHoldEnabled, float64 inputVal);

Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4204 device. You must supply a known voltage to the device and specify that voltage with **inputVal**.

Input

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelNames	const char []	The physical channel(s) to calibrate.
lowPassFreq	float64	The low pass cutoff frequency, in hertz, (6 or 10000) on the SCMP pod to calibrate.
trackHoldEnabled	bool32	Specifies whether calibrating for trackHold is enabled or disabled.
inputVal	float64	The known voltage, in volts, to use as a reference for calibration.

#### Name Type Description

# DAQmxAdjust4220Cal

int32 DAQmxAdjust4220Cal (uInt32 calHandle, const char channelNames[], float64 gain, float64 inputVal);

Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4220 device. You must supply a known voltage to the device and specify that voltage with **inputVal**.

This device needs reference signals of 0.0 volts at gains of 1, 15, 20, and 310 on a particular channel in order to perform an offset calibration for that channel. If those points are not manually supplied, they will be automatically measured internally with sample and hold enabled.

Input

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelNames	const char []	The physical channel(s) to calibrate.
gain	float64	The gain value on the SCMP pod to calibrate.
inputVal	float64	The known voltage, in volts, to use as a reference for calibration.

#### Name Type Description

# DAQmxAdjust4224Cal

int32 DAQmxAdjust4224Cal (uInt32 calHandle, const char channelNames[], float64 gain, float64 inputVal);

Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4224 device. You must supply a known voltage to the device and specify that voltage with **inputVal**.

Input

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelNames	const char []	The physical channel(s) to calibrate.
gain	float64	The gain value on the SCMP pod to calibrate.
inputVal	float64	The known voltage, in volts, to use as a reference for calibration.

#### Name Type Description
# DAQmxAdjust1102Cal

int32 DAQmxAdjust1102Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-1102 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1104Cal

int32 DAQmxAdjust1104Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the calibration constants for the SCXI-1104 module. You must measure the voltage generated by a previous call to the DAQmxSetup1104Cal function and specify the measurement with **measured output**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1112Cal

int32 DAQmxAdjust1112Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the calibration constants for the SCXI-1112 module. You must measure the voltage generated by a previous call to the DAQmxSetup1112Cal function and specify the measurement with **measOutput**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1122Cal

int32 DAQmxAdjust1122Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the calibration constants for the SCXI-1122 module. You must measure the voltage or current generated by a previous call to the DAQmxSetup1122Cal function and specify the measurement with **measOutput**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage or current measured at the output channel specified in a previous call to the DAQmxSetup1122Cal function.

#### Name Type Description

# DAQmxAdjust1124Cal

int32 DAQmxAdjust1124Cal (uInt32 calHandle, float64 measOutput);

Adjusts the calibration constants for the SCXI-1124 module. You must measure the voltage or current generated by a previous call to the DAQmxSetup1124Cal function and specify the measurement with **measOutput**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
measOutput	float64	The voltage or current measured at the output channel specified in a previous call to the DAQmxSetup1124Cal function.

#### Name Type Description

# DAQmxAdjust1125Cal

int32 DAQmxAdjust1125Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-1125 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1126Cal

int32 DAQmxAdjust1126Cal (uInt32 calHandle, float64 refFreq, float64 measOutput);

Adjusts the calibration constants for the SCXI-1126 module. You must measure the voltage generated by a previous call to the DAQmxSetup1126Cal function and specify the measurement with **measOutput**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refFreq	float64	The known frequency, in hertz, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1141Cal

int32 DAQmxAdjust1141Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-1141 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1142Cal

int32 DAQmxAdjust1142Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-1142 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description
# DAQmxAdjust1143Cal

int32 DAQmxAdjust1143Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-1143 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1502Cal

int32 DAQmxAdjust1502Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the calibration constants for the SCXI-1502 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1503Cal

int32 DAQmxAdjust1503Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the calibration constants for the SCXI-1503 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1503CurrentCal

int32 DAQmxAdjust1503CurrentCal (uInt32 calHandle, const char channelName[], float64 measOutput);

Adjusts the current calibration constants for the SCXI-1503 module.

Input
-------

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
measOutput	float64	The current measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1520Cal

int32 DAQmxAdjust1520Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-1520 module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

This device needs reference signals of 0.0 volts at gains of 1, 15, 20, and 310 on a particular channel in order to perform an offset calibration for that channel.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust1521Cal

int32 DAQmxAdjust1521Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the calibration constants for the SCXI-1521 module. You must measure the voltage generated by a previous call to the DAQmxSetup1521Cal function and specify the measurement with **measOutput**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration.
measOutput	float64	The voltage measured at the output of the module.

#### Name Type Description

# DAQmxAdjust153xCal

int32 DAQmxAdjust153xCal (uInt32 calHandle, float64 refVoltage, float64 measOutput);

Adjusts the external calibration constants for the SCXI-153x module. You must supply a known voltage to the device and specify that voltage with **refVoltage**.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
refVoltage	float64	The known voltage, in volts, to use as a reference for calibration. Both <b>refVoltage</b> and <b>measOutput</b> must be of the same measurement type, either RMS voltage or peak-to-peak voltage.
measOutput	float64	The voltage measured at the output of the module. Both <b>refVoltage</b> and <b>measOutput</b> must be of the same measurement type, either RMS voltage or peak-to-peak voltage.

#### Name Type Description

# DAQmxAdjust1540Cal

int32 DAQmxAdjust1540Cal (uInt32 calHandle, float64 refVoltage, float64 measOutput, int32 inputCalSource);

Adjusts the calibration constants for the SCXI-1540 module. You must measure the voltage generated by a previous call to the DAQmxSetup1540Cal function and specify the measurement with **measOutput**.

Input			
Name	Туре	Description	
calHandle	uInt32	A reference to the calibration session that you c function.	reated using the DAQmxInitExtCal
refVoltage	float64	The excitation RMS voltage measured from the front of the module.	
measOutput	float64	The voltage measured at the output of the modu	ıle.
inputCalSource	<b>Source</b> int32 The calibration input source selection.		
		Value	Description
		DAQmx_Val_Loopback0	Loopback the internal excitation voltage with 0 degree phase shift.
		DAQmx_Val_Loopback180	Loopback the internal excitation voltage with 180 degree phase shift.
		DAQmx_Val_Ground	Connect the channel to ground.

#### Name Type Description

# DAQmxSetup1102Cal

int32 DAQmxSetup1102Cal (uInt32 calHandle, const char channelNames[], float64 gain);

Sets the SCXI-1102 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1102Cal function.



**Note** The terminal where module output is measured will depend on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0+/- pins of the rear signal connector. Please refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelNames	const char []	The physical channel(s) to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description
# DAQmxSetup1104Cal

int32 DAQmxSetup1104Cal (uInt32 calHandle, const char channel[]);

Specifies the channel on the SCXI-1104 module for calibration. Measure calibration input/output points by supplying reference signals to the specified channel and measuring the outputs. Specify each of these points using the DAQmxAdjust1104Cal function.



**Note** The terminal where module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0+/- pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input

1		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channel	const char []	The physical channel to calibrate.

#### Name Type Description

# DAQmxSetup1112Cal

int32 DAQmxSetup1112Cal (uInt32 calHandle, const char channel[]);

Specifies the channel on the SCXI-1112 module for calibration. Measure calibration input/output points by supplying reference signals to the specified channel and measuring the outputs. Specify each of these points using the DAQmxAdjust1112Cal function.



**Note** The terminal where module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0+/- pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input

1		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channel	const char []	The physical channel to calibrate.

#### Name Type Description

# DAQmxSetup1122Cal

int32 DAQmxSetup1122Cal (uInt32 calHandle, const char channel[], float64 gain);

Sets the SCXI-1122 module to the specified gain value. Calibration I/O points can be measured by supplying reference signals to the device and measuring the outputs. Specify each of these points using the DAQmxAdjust1122Cal function.



**Note** The terminal where module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0 $\pm$  pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input

-		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channel	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1124Cal

int32 DAQmxSetup1124Cal (uInt32 calHandle, const char channelName[], int32 range, uInt32 dacValue);

Writes the specified binary value to the D/A circuitry on the specified channel at the specified range. Measure and specify the voltage or current generated in a subsequent call to the DAQmxAdjust1124Cal function.



Note Specify at least two calibration points for each channel/range being calibrated. The recommended binary data to use for voltage ranges are 0 and 4095. The recommended binary data to use for the current range is 255 and 4095.

Input

-		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
range	int32	The range to calibrate.
dacValue	uInt32	The binary number to write to the DAC circuitry.

#### Name Type Description

# DAQmxSetup1125Cal

int32 DAQmxSetup1125Cal (uInt32 calHandle, const char channelNames[], float64 gain);

Sets the SCXI-1125 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1125Cal function.



**Note** The terminal where module output is measured will depend on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0+/- pins of the rear signal connector. Please refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelNames	const char []	The physical channel(s) to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1126Cal

int32 DAQmxSetup1126Cal (uInt32 calHandle, const char channelName[], float64 upperFreqLimit);

Specifies the channel and upper frequency limit on the SCXI-1126 module for calibration. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Specify each of these points using the DAQmxAdjust1126Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
upperFreqLimit	float64	The high frequency limit in hertz, with 0 Hz as the low frequency limit, which most closely encapsulates the ranges to be calibrated.

#### Name Type Description

# DAQmxSetup1141Cal

int32 DAQmxSetup1141Cal (uInt32 calHandle, const char channelName[], float64 gain);

Sets the SCXI-1141 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1141Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1142Cal

int32 DAQmxSetup1142Cal (uInt32 calHandle, const char channelName[], float64 gain);

Sets the SCXI-1142 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1142Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1143Cal

int32 DAQmxSetup1143Cal (uInt32 calHandle, const char channelName[], float64 gain);

Sets the SCXI-1143 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1143Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description
# DAQmxSetup1502Cal

int32 DAQmxSetup1502Cal (uInt32 calHandle, const char channelName[], float64 gain);

Sets the SCXI-1502 module to the specified gain value. Calibration I/O points can be measured by supplying reference signals to the specified channel and measuring the outputs. Specify each of these points using the DAQmxAdjust1502Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1503Cal

int32 DAQmxSetup1503Cal (uInt32 calHandle, const char channelName[], float64 gain);

Sets the SCXI-1503 module to the specified gain value. Calibration I/O points can be measured by supplying reference signals to the specified channel and measuring the outputs. Specify each of these points using the DAQmxAdjust1503Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1520Cal

int32 DAQmxSetup1520Cal (uInt32 calHandle, const char channelNames[], float64 gain);

Sets the SCXI-1520 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1520Cal function. This function will also disable sample and hold on the module so that the output can be measured by external devices that cannot supply the appropriate sample and hold timing signals.



**Note** The terminal where module output is measured will depend on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0+/- pins of the rear signal connector. Please refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelNames	const char []	The physical channel(s) to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup153xCal

int32 DAQmxSetup153xCal (uInt32 calHandle, const char channelName[], float64 gain);

Sets the SCXI-153x module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust153xCal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Input		
Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
gain	float64	The gain value to calibrate.

#### Name Type Description

# DAQmxSetup1540Cal

int32 DAQmxSetup1540Cal (uInt32 calHandle, const char channel[], float64 excitationVoltage, float64 excitationFreq);

Sets the SCXI-1540 module to the specified gain value. Calibration input/output points can be measured by supplying reference signals to the specified channel and measuring the outputs. Each of these points should be specified using the DAQmxAdjust1540Cal function.



**Note** The terminal where the module output is measured depends on the configuration of the module in MAX. National Instruments recommends cabling the module to the digitizer so that the output appears on the MCH0± pins of the rear signal connector. Refer to the module user manual for more information on the routing of module output.

Name	Туре	Description
calHandle	uInt32	A reference to the calibration session that you created using the DAQmxInitExtCal function.
channelName	const char []	The physical channel to calibrate.
excitationVoltage	float64	The RMS value of the internal AC excitation voltage.
excitationFreq	float64	The frequency of the internal AC excitation voltage.

#### Name Type Description

# DAQmxDeleteSavedGlobalChan

int32 DAQmxDeleteSavedGlobalChan (const char channelName[]);

Deletes the specified  $_{global virtual channel}$  from MAX. This function does not remove the global virtual channel from  $_{tasks}$  that use it.

Input

Name Type Description

channelName const char [] Name of the global virtual channel to delete.

#### Name Type Description

# DAQmxDeleteSavedScale

int32 DAQmxDeleteSavedScale (const char scaleName[]);

Deletes the specified *custom scale* from MAX. This function does not remove the custom scale from *virtual channels* that use it.

Input
Name Type Description

scaleName const char [] Name of the custom scale to delete.

#### Name Type Description

# DAQmxDeleteSavedTask

int32 DAQmxDeleteSavedTask (const char taskName[]);

Deletes the specified  $_{task}$  from MAX. This function does not clear the copy of the task stored in memory. Use  $_{DAQmxClearTask}$  to clear the copy of the task.

 Input
 Description

 Name
 Type
 Description

 taskName
 const char []
 Name of the task to delete.

#### Name Type Description

# DAQmxSaveGlobalChan

Saves the specified local or global virtual channel to MAX as a global virtual channel. You must specify both the local or global virtual channel to save and a task that contains that channel.

Programmatically saved global virtual channels cannot be viewed in the DAQ Assistant for versions of NI-DAQ earlier than 7.4. To view a programmatically saved global virtual channel in an earlier version of NI-DAQ, first use the DAQ Assistant in NI-DAQ 7.4 or later to save the global virtual channel.

Visit the <u>DAQmx Professional Developer Tools</u> website for more information and examples of programmatically saving global virtual channels.

Input

Name	Туре	Description
taskHandle	TaskHandle	The task that contains the local or global virtual channel you want to save.
channelName	const char []	Name of the local or global virtual channel to save.
saveAs	const char []	Name to save the global virtual channel as. If you pass an empty string ("") or NULL name currently assigned to the global virtual channel will be used.
author	const char []	Name to store with the global virtual channel.
options	uInt32	Use this parameter to set certain options. You can combine options with the bitwise-C operator (' ') to set multiple options. Pass a value of zero if no options need to be set.
		Value Description

DAQmx\_Val\_Save\_Overwrite Overwrite global virtual channel of the same name if or is already saved in MAX. If you do not set this fla and a glob virtual channel of the same name is already saved in MAX, the function returns an error. DAQmx\_Val\_Save\_AllowInteractiveEditing Allow the global virtual

	channel to be edited i the DAQ Assistant. you set thi flag, the DAQ Assistant must support all global virtual channel settings.
DAQmx_Val_Save_AllowInteractiveDeletion	Allow the global virtual channel to be deleted through MAX.
#### Name Type Description

# DAQmxSaveScale

int32 DAQmxSaveScale (const char scaleName[], const char saveAs[], const char author[], uInt32 options);

Saves the specified *custom scale* to MAX.

Visit the <u>DAQmx Professional Developer Tools</u> website for more information and examples of programmatically saving global channels.

Input		
Name	Туре	Description
scaleName	const char []	Name of the custom scale to save.
saveAs	const char []	Name to save the custom scale as. If you pass an empty string ("") or NULL, the name currently assigned to the scale is used.
author	const char []	Name to store with the custom scale.
options	uInt32	Use this parameter to set certain options. You can combine options with the bitwise-OR operator (' ') to set multiple options. Pass a value of zero if no options need to be set.

DAQmx\_Val\_Save\_Overwrite

### Value

### Description

Overwrite a custom scale of the same name if one is already saved in MAX. If you do not set this flag and a custom scale of the same name is already saved in MAX, the function returns an error. Allow the custom scale to be edited in the DAQ Assistant.

DAQmx\_Val\_Save\_AllowInteractiveEditing

DAQmx\_Val\_Save\_AllowInteractiveDeletion Allow the custom scale to be deleted through MAX.

#### Name Type Description

# DAQmxSaveTask

int32 DAQmxSaveTask (TaskHandle taskHandle, const char saveAs[], const char author[], uInt32 options);

Saves the specified  $_{task}$  and any  $_{local channels}$  it contains to MAX. This function does not save global channels. Use  $_{DAQmxSaveGlobalChan}$  to save global channels.

Programmatically saved tasks cannot be viewed in the DAQ Assistant for versions of NI-DAQ earlier than 7.4. To view a programmatically saved task in an earlier version of NI-DAQ, first use the DAQ Assistant in NI-DAQ 7.4 or later to save the task.

Visit the <u>DAQmx Professional Developer Tools</u> website for more information and examples of programmatically saving global channels.

Input			
Name	Туре	Description	
taskHandle	TaskHandle	The task to save.	
saveAs	const char []	Name to save the task as. If you pass an empty string ("") or NULL, assigned to the task will be used.	the name currently
author	const char []	Name to store with the task.	
options	uInt32	Use this parameter to set certain options. You can combine options w operator (' ') to set multiple options. Pass a value of zero if no options	ith the bitwise-OR need to be set.
		Value	Description
		DAQmx_Val_Save_Overwrite	Overwrite a task of the same name if one is already saved in MAX. If you do not set this flag and a task of the same name is already saved in MAX, the function returns an error.
		DAQmx_Val_Save_AllowInteractiveEditing	Allow the task to be edited in the DAQ Assistant. If you set this flag, the DAQ

	Assistant
	must
	support all task settings.
DAQmx_Val_Save_AllowInteractiveDeletion	Allow the task to be deleted through MAX.

#### Name Type Description

# DAQmxClearTEDS

int32 DAQmxClearTEDS (const char physicalChannel[]);

Removes TEDS information from the physical channel you specify. This function temporarily overrides any TEDS configuration for the physical channel that you performed in MAX.

Input Name

Туре

Description

**physicalChannel** const char[] The name of the physical channel you want to clear.

#### Name Type Description

# DAQmxConfigureTEDS

int32 DAQmxConfigureTEDS (const char physicalChannel[], const char filePath[]);

Associates TEDS information with the physical channel you specify. If you do not specify the filename of a data sheet in the **filePath** parameter, this function attempts to find a TEDS sensor connected to the physical channel. This function temporarily overrides any TEDS configuration for the physical channel that you performed in MAX.

Input		
Name	Туре	Description
physicalChannel	const char []	The name of the physical channel you want to configure.
filePath	const char []	The path to a Virtual TEDS data sheet that you want to associate with the physical channel. If you do not specify the filename of a data sheet, this function attempts to find a TEDS sensor connected to the physical channel.

#### Name Type Description

# DAQmxWriteToTEDSFromArray

int32 DAQmxWriteToTEDSFromArray (const char physicalChannel[], uInt8 bitstream[], uInt32 arraySize, int32 basicTEDSOptions);

Writes TEDS data, stored as a 1D array of 8-bit unsigned integers, to the sensor connected to the physical channel you specify.

Input
-------

Name	Туре	Description	
physicalChannel	const char []	The name of the physical channel you want to configure. ]	
bitstream	uInt8	Represents the TEDS constructed according	bitstream to write to the sensor. This bitstream must be to the IEEE 1451.4 specification.
arraySize	uInt32	Number of bytes in th	e bitstream.
basicTEDSOptions	int32	Specifies how to hand	lle <u>basic TEDS data</u> in the bitstream.
		Value	Description
		Do Not Write	Ignore basic TEDS data.
		Write to EEPROM	Write basic TEDS data to the EEPROM, even if the sensor includes a PROM. You cannot write basic TEDS data if the PROM contains data.
		Write to PROM	Write basic TEDS data to the PROM. Any subsequent attempts to write basic TEDS data result in an error.

#### Name Type Description

# DAQmxWriteToTEDSFromFile

int32 DAQmxWriteToTEDSFromFile (const char physicalChannel[], const char filePath[], int32 basicTEDSOptions);

Writes  $_{\underline{\text{TEDS}}}$  data from a virtual TEDS file to the sensor connected to the physical channel you specify.

Input

Name	Туре	Description
physicalChannel	const char []	The name of the physical channel you want to configure.
filePath	const char []	Specifies the filename of a virtual TEDS file that contains the bitstream to write.

**basicTEDSOptions** int32 Specifies how to handle <u>basic TEDS data</u> in the bitstream.

Value	Description
Do Not Write	Ignore basic TEDS data.
Write to EEPROM	Write basic TEDS data to the EEPROM, even if the sensor includes a PROM. You cannot write basic TEDS data if the PROM contains data.
Write to PROM	Write basic TEDS data to the PROM. Any subsequent attempts to write basic TEDS data result in an error.

#### Name Type Description

# DAQmxSetAnalogPowerUpStates

int32 DAQmxSetAnalogPowerUpStates (const char deviceName[], const char channelNames[], float64 state, int32 channelType, ...);

Updates the states that analog physical channels on a device are set to when the device powers up or when the device is reset. Power-up states are stored in EEPROMs that you can write to only a limited number of times. Therefore, you should use this function as infrequently as possible. This function writes to the EEPROM only if a setting you request is different from the one currently stored on the EEPROM. This function writes power-up states in sequential order. Therefore, if a physical channel has multiple entries, the last entry is used.

Input

Name	Туре	Description		
deviceName	const char []	ne name of the device, as configured in Measurement & utomation Explorer (MAX), to which this operation applies.		
channelNames	const char []	The physical channel to modify. You can <u>specify</u> <u>channels</u> .	e physical channel to modify. You can <u>specify a list or range of</u> <u>mels</u> .	
state	float64	The power-up state to set for the channel(s) in	channelNames.	
channelType	int32	The channel type for the channel(s) in <b>channel</b>	Names.	
		Value	Description	
		DAQmx_Val_ChannelVoltage	Voltage output. You can set voltage power-up states only for physical channels that support voltage output.	
		DAQmx_Val_ChannelCurrent	Current output. You can set current power-up states only for physical channels that support current output.	
moreChannelsStatesAndTypes	any type (passed by value)	Combinations of additional channels and states channels to when the device powers up or whe You must pass NULL at the end of the argument If you do not want to pass additional channels function call can be similar to the following ex	s and types to set the n the device is reset nt list. and states, the ample:	

DAQmxSetAnalogPowerUpStates ("Dev1", "Dev1/ao0", 0.0,

DAQmx\_Val\_ChannelVoltage, NULL); If you pass additional channels and states, the function call can be similar to the following example: DAQmxSetAnalogPowerUpStates ("Dev1", "Dev1/ao0", 0.0, DAQmx\_Val\_ChannelVoltage, "Dev1/ao1", 1.0, DAQmx\_Val\_ChannelCurrent, NULL);

#### Name Type Description

# DAQmxSetDigitalLogicFamilyPowerUpState

Sets the digital logic family to use when the device powers up.

Input				
Name	Туре	Description		
deviceName	const char []	The name of the device, as configured in Measurement & Automation Explorer (MAX), to which this operation applies.		
logicFamily int32		Specifies the logic family to set the device to when it powers up. A logic family corresponds to voltage thresholds that are compatible with a group of voltage standards. Refer to device documentation for information on the logic high and logic low voltages for these logic families.		
		Value	Description	
		DAQmx_Val_2point5V	2.5 V (compatible with CMOS signals)	
		DAQmx_Val_3point3V	3.3 V (compatible with LVTTL and LVCMOS signals)	
		DAQmx_Val_5V	5 V (compatible with TTL and CMOS signals)	

#### Name Type Description
# DAQmxSetDigitalPowerUpStates

int32 DAQmxSetDigitalPowerUpStates (const char deviceName[], const char channelNames[], int32 state, ...);

## Purpose

Updates the state that digital physical channels on static DIO devices are set to when the device powers up or when the device is reset. Power-up states are stored in EEPROMs that you can write to only a limited number of times. Therefore, you should use this function as infrequently as possible. This function writes to the EEPROM only if a setting you request is different from the one currently stored on the EEPROM. This function writes power-up states in sequential order. Therefore, if a physical channel has multiple entries, the last entry is used.

### **Parameters**

Input

Name	Туре	Description	
deviceName	const char []	The name of the device, as configured Explorer (MAX), to which this operat	in Measurement & Automation ion applies.
channelNames	const char []	The digital line or port to modify. You cannot set power-up states for dedicated digital input lines. You can <u>specify a list or range of channels</u> .	
state	int32	The power-up state to set for the channel(s) in <b>channelNames</b> .	
		Value	Description
		DAQmx_Val_High	High logic
		DAQmx_Val_Low	Low logic
		DAQmx_Val_Tristate	High-impedance state. You can select this state only on devices with bidirectional ports, and you can select it only for entire ports. You cannot select this state for dedicated digital output lines.
moreChannelsAndStates	any type (passed by value)	Pairs of additional channels and the states to set the channels to when the device powers up or when the device is reset. You must pass NULL at the end of the argument list. If you do not want to pass additional channels and states, the function call can be similar to the following example: DAQmxSetDigitalPowerUpStates ("Dev1", "Dev1/do0", DAQmx_Val_High, NULL); If you pass additional channels and states, the function call can be similar to the following example: DAQmxSetDigitalPowerUpStates ("Dev1", "Dev1/do0", DAQmxSetDigitalPowerUpStates ("Dev1", "Dev1/do0", DAQmxSetDigitalPowerUpStates ("Dev1", "Dev1/do0", DAQmx_Val_High, "Dev1/do1", DAQmx_Val_Tristate, NULL);	

### **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A positive value indicates a warning. A negative value indicates an error.

# DAQmxGetErrorString

int32 DAQmxGetErrorString (int32 errorCode, char errorString[], uInt32 bufferSize);

# Purpose

Converts the error number returned by an NI-DAQmx function into a meaningful error message.

If you pass in a valid value for **errorString** and its **bufferSize**, this function returns as much of the available data as possible.

## Parameters

Input		
Name	Туре	Description
errorCode	int32	An error code or warning returned by one of the NI-DAQmx Library functions.
bufferSize	uInt32	The size, in bytes, of the buffer passed in the <b>errorString</b> . If you pass 0, this function returns the number of bytes you need to allocate.
Output		
errorString	char []	The meaningful error message for the error number. If you pass NULL, this function returns the number of bytes you need to allocate.

### **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error.

If you pass in a valid value for **errorString** and its **bufferSize**, this function returns as much of the available data as possible.

# DAQmxGetExtendedErrorInfo

int32 DAQmxGetExtendedErrorInfo (char errorString[], uInt32 bufferSize);

# Purpose

Returns dynamic, specific error information. This function is valid only for the last function that failed; additional NI-DAQmx calls may invalidate this information.

If you pass valid values for **errorString** and **bufferSize**, this function returns as much of the available data as possible.

# **Parameters**

Input		
Name	Туре	Description
bufferSize	uInt32	The size, in bytes, of <b>errorString</b> . If you pass 0, this function returns the number of bytes you need to allocate.
Output		
Name	Туре	Description
errorString	char []	Dynamic error information. If you pass NULL, this function returns the number of bytes you need to allocate.

### **Return Value**

#### Name Type Description

**status** int32 The error code returned by the function in the event of an error or warning. A value of 0 indicates success. A negative value indicates an error.

If you pass in a valid value for **errorString** and its **bufferSize**, this function returns as much of the available data as possible.