

APPENDIX D

A/D INTERFACE CARD DRIVER FUNCTIONS

DAQmxCreateTask

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

Purpose

Creates a [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.

Parameters

Input

Name	Type	Description
		Name assigned to the task.

taskName const char []



Note This name may be changed internally. Call DAQmxGetTaskName to verify whether the name was changed during creation.

Output

Name	Type	Description
taskHandle	TaskHandle *	A reference to the task created in this function.

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.

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 [DAQmxCreateAIVoltageChanWithExcit](#).

Parameters

Input

Name	Type	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 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 nameToAssignToChannel , 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.

terminalConfig	int32	The input terminal configuration for the channel.
-----------------------	-------	---

Value

DAQmx_Val_Cfg_Default (-1)

DAQmx_Val_RSE

Description

At run time, NI-DAQmx chooses the [default terminal configuration](#) for the channel.

[Referenced single-ended mode](#)

		DAQmx_Val_NRSE	Nonreferenced single-ended mode
		DAQmx_Val_Diff	Differential mode
		DAQmx_Val_PseudoDiff	Pseudodifferential mode
minVal	float64	The minimum value , in units , that you expect to measure.	
maxVal	float64	The maximum value , in units , that you expect to measure.	
units	int32	The units to use to return the voltage measurements.	
		Name	Description
		DAQmx_Val_Volts	volts
		DAQmx_Val_FromCustomScale	Units a custom scale specifies. Use customScaleName to specify a custom scale.
customScaleName	const char []	The name of a custom scale 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.	

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 [Sample Clock](#), the rate of the Sample Clock, and the number of samples to acquire or generate.

Parameters

Input

Name	Type	Description								
taskHandle	TaskHandle	The task used in this function.								
source	const char []	The source terminal of the Sample Clock. To use the internal clock of the device, use NULL or use OnboardClock.								
rate	float64	The sampling rate in samples per second. If you use an external source for the Sample Clock, set this value to the maximum expected rate of that clock.								
activeEdge	int32	Specifies on which edge of the clock to acquire or generate samples. <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>DAQmx_Val_Rising</td> <td>Acquire or generate samples on the rising edges of the Sample Clock.</td> </tr> <tr> <td>DAQmx_Val_Falling</td> <td>Acquire or generate samples on the falling edges of the Sample Clock.</td> </tr> </tbody> </table>	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.		
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.									
sampleMode	int32	Specifies whether the task acquires or generates samples continuously or if it acquires or generates a finite number of samples. <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>DAQmx_Val_FiniteSamps</td> <td>Acquire or generate a finite number of samples.</td> </tr> <tr> <td>DAQmx_Val_ContSamps</td> <td>Acquire or generate samples until you stop the task.</td> </tr> <tr> <td>DAQmx_Val_HWTimedSinglePoint</td> <td>Acquire or generate samples continuously using hardware timing without a buffer. Hardware timed single point</td> </tr> </tbody> </table>	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. Hardware timed single point
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. Hardware timed single point									

sample mode is supported only for the sample clock and change detection timing types.

sampsPerChanToAcquire uInt64

The number of samples to acquire or generate for each channel in the task if **sampleMode** is DAQmx_Val_FiniteSamps. If **sampleMode** is DAQmx_Val_ContSamps, NI-DAQmx uses this value to [determine the buffer size](#).

Return Value

Name
status

Type
int32

Description

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.

DAQmxDisableStartTrig

```
int32 DAQmxDisableStartTrig (TaskHandle taskHandle);
```

Purpose

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

Parameters

Input

Name	Type	Description
taskHandle	TaskHandle	The task used in this function.

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.

DAQmxStartTask

```
int32 DAQmxStartTask (TaskHandle taskHandle);
```

Purpose

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.

Parameters

Input

Name	Type	Description
taskHandle	TaskHandle	The task to start.

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.

DAQmxReadAnalogF64

int32 DAQmxReadAnalogF64 (TaskHandle taskHandle, int32 numSampsPerChan, float64 timeout, bool32 fillMode, float64 readArray[], uInt32 arraySizeInSamps, int32 *sampsPerChanRead, bool32 *reserved);

Purpose

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

Parameters

Input

Name	Type	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 readArray does not contain enough space, this function returns as many samples as fit in readArray .

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 interleaved .
		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.
<i>Output</i>		
Name	Type	Description
readArray	float64 []	The array to read samples into, organized according to fillMode .
sampsPerChanRead	int32 *	The actual number of samples read from each channel.

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.

Interleaving

Interleaved samples prioritize samples before channels, such that the array lists the first sample from every channel in the task, then the second sample from every channel, up to the last sample from every channel.

Channel 0—Sample 1
Channel 1—Sample 1
Channel 2—Sample 1
Channel 0—Sample 2
Channel 1—Sample 2
Channel 2—Sample 2
...
Channel 0—Sample N
Channel 1—Sample N
Channel 2—Sample N

Non-interleaved samples prioritize channels before samples, such that the array lists all samples from the first channel in the task, then all samples from the second channel, up to all samples from the last channel.

Channel 0—Sample 1
Channel 0—Sample 2
...
Channel 0—Sample N
Channel 1—Sample 1
Channel 1—Sample 2
...
Channel 1—Sample N
Channel 2—Sample 1
Channel 2—Sample 2
...
Channel 2—Sample N

DAQmxStopTask

```
int32 DAQmxStopTask (TaskHandle taskHandle);
```

Purpose

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.

Parameters

Input

Name	Type	Description
taskHandle	TaskHandle	The task to stop.

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.