

WaveformChassis Project

Generated by Doxygen 1.8.2

Mon Nov 19 2012 11:30:25

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	AnalogOutput.AnalogOutput Class Reference	5
3.1.1	Detailed Description	6
3.1.2	Constructor & Destructor Documentation	7
3.1.2.1	__init__	7
3.1.2.2	__del__	7
3.1.3	Member Function Documentation	7
3.1.3.1	close	7
3.1.3.2	createSineTestBuffer	7
3.1.3.3	createTestBuffer	7
3.1.3.4	getClkSource	7
3.1.3.5	getNumChannels	8
3.1.3.6	getSampleRate	8
3.1.3.7	getSamplesPerChannel	8
3.1.3.8	getStartTriggerSource	8
3.1.3.9	init	8
3.1.3.10	setClkSource	8
3.1.3.11	setSampleRate	9
3.1.3.12	setSamplesPerChannel	9
3.1.3.13	setStartTriggerSource	9
3.1.3.14	start	9
3.1.3.15	stop	9
3.1.3.16	waitUntilDone	9
3.1.3.17	writeToBuffer	10
3.1.4	Member Data Documentation	10
3.1.4.1	estAcqTime	10

3.1.4.2	initialized	10
3.1.4.3	loops	10
3.1.4.4	mode	10
3.1.4.5	startTriggerSyncCard	10
3.1.4.6	status	10
3.1.4.7	taskRef	11
3.1.4.8	triggerType	11
3.1.5	Property Documentation	11
3.1.5.1	clkSource	11
3.1.5.2	numChannels	11
3.1.5.3	sampleRate	11
3.1.5.4	samplesPerChannel	11
3.1.5.5	startTriggerSource	11
3.2	chassisConfigParser.chassisConfigParser Class Reference	11
3.2.1	Detailed Description	12
3.2.2	Member Function Documentation	12
3.2.2.1	createDefaultFile	12
3.2.2.2	read	12
3.2.2.3	write	12
3.3	DigitalOutput.DigitalOutput Class Reference	13
3.3.1	Detailed Description	14
3.3.2	Constructor & Destructor Documentation	14
3.3.2.1	__init__	14
3.3.2.2	__del__	14
3.3.3	Member Function Documentation	15
3.3.3.1	close	15
3.3.3.2	createTestBuffer	15
3.3.3.3	getClkSource	15
3.3.3.4	getNumChannels	15
3.3.3.5	getNumLines	15
3.3.3.6	getSampleRate	15
3.3.3.7	getSamplesPerChannel	16
3.3.3.8	init	16
3.3.3.9	setClkSource	16
3.3.3.10	setSampleRate	16
3.3.3.11	setSamplesPerChannel	16
3.3.3.12	start	16
3.3.3.13	stop	17
3.3.3.14	waitUntilDone	17
3.3.3.15	writeToBuffer	17

3.3.4	Member Data Documentation	17
3.3.4.1	initialized	17
3.3.4.2	loops	17
3.3.4.3	mode	17
3.3.4.4	status	18
3.3.4.5	taskRef	18
3.3.5	Property Documentation	18
3.3.5.1	clkSource	18
3.3.5.2	sampleRate	18
3.3.5.3	samplesPerChannel	18
3.4	eMapParser.eMapParser Class Reference	18
3.4.1	Detailed Description	19
3.4.2	Member Function Documentation	19
3.4.2.1	read	19
3.5	fileParser.fileParser Class Reference	19
3.5.1	Detailed Description	20
3.5.2	Constructor & Destructor Documentation	20
3.5.2.1	__init__	20
3.5.3	Member Function Documentation	20
3.5.3.1	close	20
3.5.3.2	getNumLines	20
3.5.3.3	open	21
3.5.4	Member Data Documentation	21
3.5.4.1	comments	21
3.5.4.2	empty	21
3.5.4.3	fileObj	21
3.5.4.4	meta	21
3.5.5	Property Documentation	21
3.5.5.1	totalLines	21
3.6	fileParser.fileParserError Class Reference	21
3.6.1	Detailed Description	22
3.6.2	Constructor & Destructor Documentation	22
3.6.2.1	__init__	22
3.6.3	Member Function Documentation	22
3.6.3.1	__str__	22
3.6.4	Member Data Documentation	22
3.6.4.1	msg	22
3.7	itfParser.itfParser Class Reference	23
3.7.1	Detailed Description	24
3.7.2	Constructor & Destructor Documentation	24

3.7.2.1	<code>__init__</code>	24
3.7.3	Member Function Documentation	24
3.7.3.1	<code>appendline</code>	24
3.7.3.2	<code>eMapRead</code>	24
3.7.3.3	<code>eMapReadLine</code>	24
3.7.3.4	<code>eMapReadLines</code>	25
3.7.3.5	<code>read</code>	25
3.7.3.6	<code>readline</code>	25
3.7.3.7	<code>readlines</code>	25
3.7.4	Member Data Documentation	25
3.7.4.1	<code>comments</code>	25
3.7.4.2	<code>eMapFilePath</code>	26
3.7.4.3	<code>empty</code>	26
3.7.4.4	<code>meta</code>	26
3.7.4.5	<code>tableHeader</code>	26
3.8	DAQmxUtility.Mode Class Reference	26
3.8.1	Detailed Description	26
3.8.2	Member Data Documentation	26
3.8.2.1	<code>Continuous</code>	26
3.8.2.2	<code>Finite</code>	27
3.8.2.3	<code>Static</code>	27
3.9	niSyncError.niSyncError Class Reference	27
3.9.1	Detailed Description	27
3.9.2	Constructor & Destructor Documentation	27
3.9.2.1	<code>__init__</code>	27
3.9.3	Member Function Documentation	28
3.9.3.1	<code>__str__</code>	28
3.9.4	Member Data Documentation	28
3.9.4.1	<code>code</code>	28
3.9.4.2	<code>msg</code>	28
3.10	pCounter.pCounter Class Reference	28
3.10.1	Detailed Description	29
3.10.2	Constructor & Destructor Documentation	30
3.10.2.1	<code>__init__</code>	30
3.10.2.2	<code>__del__</code>	30
3.10.3	Member Function Documentation	30
3.10.3.1	<code>close</code>	30
3.10.3.2	<code>init</code>	30
3.10.3.3	<code>initFromFile</code>	30
3.10.3.4	<code>measure</code>	31

3.10.3.5	read	31
3.10.3.6	start	31
3.10.3.7	stop	31
3.10.4	Member Data Documentation	31
3.10.4.1	clockCntrTask	31
3.10.4.2	clockCounter	31
3.10.4.3	clockSourceTerm	32
3.10.4.4	edgeCntrTask	32
3.10.4.5	edgeCntrTerm	32
3.10.4.6	edgeCounter	32
3.10.4.7	enableStartTrigger	32
3.10.4.8	timeout	32
3.10.4.9	triggerSource	32
3.10.5	Property Documentation	32
3.10.5.1	acqTime	32
3.10.5.2	binTime	32
3.10.5.3	sampleRate	33
3.10.5.4	samples	33
3.11	Timing.Timing Class Reference	33
3.11.1	Detailed Description	34
3.11.2	Constructor & Destructor Documentation	34
3.11.2.1	__init__	34
3.11.2.2	__del__	34
3.11.3	Member Function Documentation	34
3.11.3.1	close	34
3.11.3.2	init	34
3.11.3.3	sendSoftwareTrigger	35
3.11.4	Member Data Documentation	35
3.11.4.1	ddsFreq	35
3.11.4.2	divisor	35
3.11.4.3	initialized	35
3.11.4.4	pxiStarSlots	35
3.11.4.5	resourceName	35
3.11.4.6	sampleRate	35
3.11.4.7	session	35
3.11.4.8	status	35
3.12	DAQmxUtility.TriggerType Class Reference	36
3.12.1	Detailed Description	36
3.12.2	Member Data Documentation	36
3.12.2.1	Hardware	36

3.12.2.2	Software	36
3.13	WaveformChassis.WaveformChassis Class Reference	36
3.13.1	Detailed Description	38
3.13.2	Constructor & Destructor Documentation	38
3.13.2.1	__init__	38
3.13.3	Member Function Documentation	38
3.13.3.1	close	38
3.13.3.2	createAoSineBuffer	38
3.13.3.3	createDoTestBuffer	38
3.13.3.4	getNumAoChannels	39
3.13.3.5	getNumDoChannels	39
3.13.3.6	init	39
3.13.3.7	initFromFile	39
3.13.3.8	start	39
3.13.3.9	stop	39
3.13.3.10	waitUntilDone	40
3.13.3.11	writeAoBuffer	40
3.13.3.12	writeDoBuffer	40
3.13.3.13	writeStartStop	40
3.13.4	Member Data Documentation	40
3.13.4.1	clkSource	40
3.13.4.2	gens	40
3.13.4.3	loops	41
3.13.4.4	mode	41
3.13.4.5	sampleRate	41
3.13.4.6	samplesPerChannel	41
3.13.4.7	startTriggerSource	41
3.13.4.8	timing	41
3.13.4.9	triggerType	41
3.13.4.10	useTiming	41
3.14	WaveformGenerator.WaveformGenerator Class Reference	42
3.14.1	Detailed Description	43
3.14.2	Constructor & Destructor Documentation	43
3.14.2.1	__init__	43
3.14.3	Member Function Documentation	43
3.14.3.1	close	43
3.14.3.2	init	43
3.14.3.3	start	43
3.14.3.4	stop	43
3.14.3.5	waitUntilDone	44

3.14.3.6	writeAoBuffer	44
3.14.3.7	writeDoBuffer	44
3.14.4	Member Data Documentation	44
3.14.4.1	ao	44
3.14.4.2	clkSource	44
3.14.4.3	do	44
3.14.4.4	loops	44
3.14.4.5	mode	44
3.14.4.6	sampleRate	45
3.14.4.7	samplesPerChannel	45
3.14.4.8	startTriggerSource	45
3.14.4.9	triggerType	45
3.14.4.10	useAo	45
3.14.4.11	useDo	45
3.15	WaveformGenerator.writeStartStop Class Reference	45
 Index		 46

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Exception	
fileParser.fileParserError	21
niSyncError.niSyncError	27
object	
AnalogOutput.AnalogOutput	5
DAQmxUtility.Mode	26
DAQmxUtility.TriggerType	36
DigitalOutput.DigitalOutput	13
fileParser.fileParser	19
eMapParser.eMapParser	18
itfParser.itfParser	23
pCounter.pCounter	28
Timing.Timing	33
WaveformChassis.WaveformChassis	36
WaveformGenerator.WaveformGenerator	42
Thread	
WaveformGenerator.writeStartStop	45
SafeConfigParser	
chassisConfigParser.chassisConfigParser	11

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AnalogOutput.AnalogOutput	
This class will generate analog outputs using pyDAQmx	5
chassisConfigParser.chassisConfigParser	
This class can read and write a WaveformChassis configuration file	11
DigitalOutput.DigitalOutput	
This class will generate digital outputs using pyDAQmx	13
eMapParser.eMapParser	
This class will parse the electrode map file	18
fileParser.fileParser	
This class is the parent class of all file parsers	19
fileParser.fileParserError	
This is a class that creates a specific exception for the fileParser class	21
itfParser.itfParser	
This class will parse through an itf file	23
DAQmxUtility.Mode	
This class defines the mode variable which is used as an enumerated typedef	26
niSyncError.niSyncError	
This is a class to handle errors generated by the niSync python module	27
pCounter.pCounter	
This class will count edges of a ditial signal at a sample clock rate specified, and has capability of being triggered to start	28
Timing.Timing	
This class will generate perform the timing and synchronization required to synchronize all cards on the PXI backplane	33
DAQmxUtility.TriggerType	
This class defines the trigger type variable which is used as an enumerated typedef	36
WaveformChassis.WaveformChassis	
This class contains a list of WaveformGenerator objects and a Timing object	36
WaveformGenerator.WaveformGenerator	
This class contains an AnalogOutput and DigitalOutput object	42
WaveformGenerator.writeStartStop	45

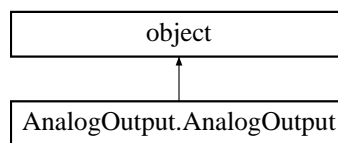
Chapter 3

Class Documentation

3.1 AnalogOutput.AnalogOutput Class Reference

This class will generate analog outputs using pyDAQmx.

Inheritance diagram for AnalogOutput.AnalogOutput:



Public Member Functions

- def `__init__`
This function is a constructor for the [AnalogOutput](#) class.
- def `init`
Initializes the analog outputs based on the object's configuration.
- def `getSamplesPerChannel`
This function returns the samples per channel configured in the DAQmx Task.
- def `setSamplesPerChannel`
This function sets the samples per channel in the DAQmx Task.
- def `getClkSource`
This function returns the sample clock source configured in the DAQmx Task.
- def `setClkSource`
This function sets the sample clock source in the DAQmx Task.
- def `getStartTriggerSource`
This function return the start trigger source configured in the DAQmx Task.
- def `setStartTriggerSource`
This function sets the start trigger source in the DAQmx Task.
- def `getNumChannels`
This function returns the number of channels configured in the DAQmx Task.
- def `getSampleRate`
This function returns the sample rate configured in the DAQmx Task.
- def `setSampleRate`
This function sets the sample rate in the DAQmx Task.
- def `createTestBuffer`

- This function returns a 1D numpy array of samples with random voltages.*
- def [createSineTestBuffer](#)
This function returns a 1D numpy array of sine waves.
- def [writeToBuffer](#)
This function writes the specified values into the buffer.
- def [start](#)
This function starts the analog output generation.
- def [waitUntilDone](#)
This functions waits for the analog output generation to complete.
- def [stop](#)
This function stops the analog output generation.
- def [close](#)
This function will close connection to the analog ouput device and channels.
- def [__del__](#)
This is the destructor for the [AnalogOutput](#) Class.

Public Attributes

- [taskRef](#)
The DAQmx task reference.
- [initialized](#)
This is a boolean that is true when the DAQmx task has been initialized.
- [status](#)
This is the status of the DAQmx task.
- [startTriggerSyncCard](#)
This is the start trigger terminal of the NI-Sync card.
- [mode](#)
This is the mode of operation for the analog outputs.
- [triggerType](#)
The trigger type for the analog outputs.
- [loops](#)
The number of times to iterate over a Finite number of samples.
- [estAcqTime](#)
The estimated time to generate the samples for a Finite generation.

Properties

- [samplesPerChannel](#)
This is the number of samples per channel that will be generated in Finite mode.
- [clkSource](#) property([getClkSource](#), [setClkSource](#), [_delClkSource](#))
This is the sample clock source terminal.
- [startTriggerSource](#) property([getStartTriggerSource](#), [setStartTriggerSource](#), [_delStartTriggerSource](#))
This is the start trigger source terminal.
- [numChannels](#) property([getNumChannels](#))
This is the number of channels configured in the task.
- [sampleRate](#) property([getSampleRate](#), [setSampleRate](#), [_delSampleRate](#))
This is the sample rate of the analog output.

3.1.1 Detailed Description

This class will generate analog outputs using pyDAQmx.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 `def AnalogOutput.AnalogOutput.__init__(self)`

This function is a constructor for the [AnalogOutput](#) class.

It creates the internal variables required to perform functions within the class. This function does not initialize any hardware.

3.1.2.2 `def AnalogOutput.AnalogOutput.__del__(self)`

This is the destructor for the [AnalogOutput](#) Class.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3 Member Function Documentation

3.1.3.1 `def AnalogOutput.AnalogOutput.close(self)`

This function will close connection to the analog output device and channels.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.2 `def AnalogOutput.AnalogOutput.createSineTestBuffer(self)`

This function returns a 1D numpy array of sine waves.

The returned value is intended to be used to write samples to the buffer with the [writeToBuffer\(\)](#) method.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.3 `def AnalogOutput.AnalogOutput.createTestBuffer(self, numChannels = 0)`

This function returns a 1D numpy array of samples with random voltages.

The returned value is intended to be used to write samples to the buffer with the [writeToBuffer\(\)](#) method.

Parameters

<i>self</i>	The object pointer.
<i>numChannels</i>	The number of channels to generate. If this parameter is not provided, Then the function will generate the number of channels configured in the analog output task.

3.1.3.4 `def AnalogOutput.AnalogOutput.getCikSource(self)`

This function returns the sample clock source configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.5 def AnalogOutput.AnalogOutput.getNumChannels (*self*)

This function returns the number of channels configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.6 def AnalogOutput.AnalogOutput.getSampleRate (*self*)

This function returns the sample rate configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.7 def AnalogOutput.AnalogOutput.getSamplesPerChannel (*self*)

This function returns the samples per channel configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.8 def AnalogOutput.AnalogOutput.getStartTriggerSource (*self*)

This function return the start trigger source configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.9 def AnalogOutput.AnalogOutput.init (*self*, *physicalChannel*)

Initializes the analog outputs based on the object's configuration.

Parameters

<i>self</i>	The object pointer.
<i>physicalChannel</i>	A string representing the device and analog output channels. Example Value: "PXI1Slot3/ao0-:7"

3.1.3.10 def AnalogOutput.AnalogOutput.setClkSource (*self*, *value*)

This function sets the sample clock source in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The string value for the clock source terminal.

3.1.3.11 def AnalogOutput.AnalogOutput.setSampleRate (*self*, *value*)

This function sets the sample rate in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The value of the sample rate.

3.1.3.12 def AnalogOutput.AnalogOutput.setSamplesPerChannel (*self*, *value*)

This function sets the samples per channel in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The value to set the samples per channel.

3.1.3.13 def AnalogOutput.AnalogOutput.setStartTriggerSource (*self*, *value*)

This function sets the start trigger source in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The string value of the start trigger source. Example value: "\\PXI1Slot3\\PFI0"

3.1.3.14 def AnalogOutput.AnalogOutput.start (*self*)

This function starts the analog output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.15 def AnalogOutput.AnalogOutput.stop (*self*)

This function stops the analog output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.16 def AnalogOutput.AnalogOutput.waitUntilDone (*self*)

This function waits for the analog output generation to complete.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.1.3.17 `def AnalogOutput.AnalogOutput.writeToBuffer (self, data)`

This function writes the specified values into the buffer.

Parameters

<i>self</i>	The object pointer.
<i>data</i>	This is a 1D 64-bit floating point numpy array that contains data for each channel. Channels are non-interleaved (channel1 n-samples then channel2 n-samples).

3.1.4 Member Data Documentation

3.1.4.1 `AnalogOutput.AnalogOutput.estAcqTime`

The estimated time to generate the samples for a Finite generation.

Once the input buffer of the analog input is configured, the amount of time it takes to generate the voltages in the buffer can be estimated. This is a function of the sample rate and the number of samples per channel. (This attribute is for internal use only. This attribute may not return an accurate value.)

3.1.4.2 `AnalogOutput.AnalogOutput.initialized`

This is a boolean that is true when the DAQmx task has been initialized.

3.1.4.3 `AnalogOutput.AnalogOutput.loops`

The number of times to iterate over a Finite number of samples.

This value is only useful in the "Finite" mode. It is the number of times that a sequence of voltages will be looped. The default is always 1.

3.1.4.4 `AnalogOutput.AnalogOutput.mode`

This is the mode of operation for the analog outputs.

There are currently three modes available. Static mode is where one static voltage is set with no need for a sample clock. Finite mode is where a finite number of voltages will be set at a sample clock rate. Continuous mode is where a sequence of voltages are generated at a sample rate and then repeated until the [stop\(\)](#) method is called.

3.1.4.5 `AnalogOutput.AnalogOutput.startTriggerSyncCard`

This is the start trigger terminal of the NI-Sync card.

Setting this value will make sure that the start trigger will be propagated through the PXI backplane. If there is no sync card needed leave the value default.

3.1.4.6 `AnalogOutput.AnalogOutput.status`

This is the status of the DAQmx task.

A value greater than 0 means that an error has occurred. When the status is greater than 0 an error should be reported by the class.

3.1.4.7 AnalogOutput.AnalogOutput.taskRef

The DAQmx task reference.

3.1.4.8 AnalogOutput.AnalogOutput.triggerType

The trigger type for the analog outputs.

There are currently two trigger types - "Software" and "Hardware." The "Software" mode means that analog output channels are not synchronized. While "Hardware" means that analog output channels are synchronized to a start trigger. The startTriggerSouce attribute must be configured appropriately.

3.1.5 Property Documentation

3.1.5.1 AnalogOutput.AnalogOutput.clkSource property(getClkSource, setClkSource, _delClkSource) [static]

This is the sample clock source terminal.

It can be set to an internal clock or external clock such as a PFI line i.e. "/PXI1Slot3/PFI15."

3.1.5.2 AnalogOutput.AnalogOutput.numChannels property(getNumChannels) [static]

This is the number of channels configured in the task.

3.1.5.3 AnalogOutput.AnalogOutput.sampleRate property(getSampleRate, setSampleRate, _delSampleRate) [static]

This is the sample rate of the analog output.

3.1.5.4 AnalogOutput.AnalogOutput.samplesPerChannel [static]

Initial value:

```
1 property(getSamplesPerChannel, setSamplesPerChannel,
2          _delSamplesPerChannel)
```

This is the number of samples per channel that will be generated in Finite mode.

3.1.5.5 AnalogOutput.AnalogOutput.startTriggerSource property(getStartTriggerSource, setStartTriggerSource, _delStartTriggerSource) [static]

This is the start trigger source terminal.

The software ignores this value when the triggerType is set to "Software". Otherwise when the triggerType is "Hardware," this terminal is used to start analog generation. Example Value: "/PXI1Slot3/PFI0"

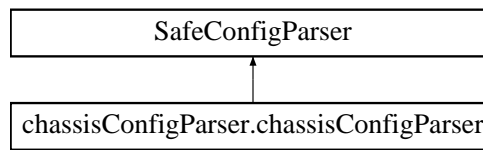
The documentation for this class was generated from the following file:

- AnalogOutput.py

3.2 chassisConfigParser.chassisConfigParser Class Reference

This class can read and write a WaveformChassis configuration file.

Inheritance diagram for chassisConfigParser.chassisConfigParser:



Public Member Functions

- def [write](#)
The function writes and creates a configuration file based on the input parameters.
- def [createDefaultFile](#)
This function will create a default file with default values at the location specified by the filePath parameter.
- def [read](#)
This function will return the information read from the configuration file.
- def **readCntrSection**

Public Attributes

- **aoChannels**
- **doChannels**
- **niSyncDev**
- **edgeCounter**
- **clockCounter**

3.2.1 Detailed Description

This class can read and write a WaveformChassis configuration file.

3.2.2 Member Function Documentation

3.2.2.1 def chassisConfigParser.chassisConfigParser.createDefaultFile (self, filePath)

This function will create a default file with default values at the location specified by the filePath parameter.

Parameters

<i>filePath</i>	The location of the file to be created.
-----------------	---

3.2.2.2 def chassisConfigParser.chassisConfigParser.read (self, filePath)

This function will return the information read from the configuration file.

3.2.2.3 def chassisConfigParser.chassisConfigParser.write (self, filePath, data, niSyncDev, edgeCounter, clockCounter)

The function writes and creates a configuration file based on the input parameters.

Parameters

<i>self</i>	The object pointer.
<i>filePath</i>	The location of the file to be created
<i>data</i>	A dictionary containing the analog output and digital output channel names.
<i>niSyncDev</i>	The name of the NI-Sync card.

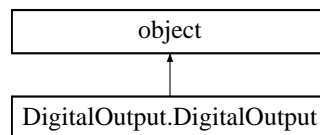
The documentation for this class was generated from the following file:

- chassisConfigParser.py

3.3 DigitalOutput.DigitalOutput Class Reference

This class will generate digital outputs using pyDAQmx.

Inheritance diagram for DigitalOutput.DigitalOutput:



Public Member Functions

- def [getSampleRate](#)
This function returns the sample rate configured in the DAQmx Task.
- def [setSampleRate](#)
This function sets the sample rate in the DAQmx Task.
- def [getSamplesPerChannel](#)
This function returns the samples per channel configured in the DAQmx Task.
- def [setSamplesPerChannel](#)
This function sets the samples per channel in the DAQmx Task.
- def [getClkSource](#)
This function returns the sample clock source configured in the DAQmx Task.
- def [setClkSource](#)
This function sets the sample clock source in the DAQmx Task.
- def [__init__](#)
This function is a constructor for the [DigitalOutput](#) class.
- def [init](#)
Initialize the digital outputs based on the object's configuration.
- def [createTestBuffer](#)
This function returns a random 1D numpy array of samples for writing the buffer of digital output channels.
- def [getNumLines](#)
This function returns the number of digital lines configured in the DAQmx Task.
- def [getNumChannels](#)
This function returns the number of digital channels configured in the DAQmx Task.
- def [writeToBuffer](#)
This function writes the specified values into the buffer.
- def [writeStatic](#)
- def [start](#)
This function starts the digital output generation.

- def [waitUntilDone](#)
This functions waits for the digital output generation to complete.
- def [stop](#)
This function stops the digital output generation.
- def [close](#)
This function will close connection to the digital ouput device and channels.
- def [__del__](#)
This is the destructor for the [DigitalOutput](#) Class.

Public Attributes

- [status](#)
This is the status of the DAQmx task.
- [taskRef](#)
The DAQmx task reference.
- [initialized](#)
This is a boolean that is true when the DAQmx task has been initialized.
- [mode](#)
This is the mode of operation for the digital outputs.
- [loops](#)
The number of time to iterate over a Finite number of samples.

Properties

- [sampleRate](#)
This is the sample rate of the digital output.
- [samplesPerChannel](#)
This is the number of samples per channel that will be generated in Finite mode.
- [clkSource](#)
This is the sample clock source terminal.

3.3.1 Detailed Description

This class will generate digital outputs using pyDAQmx.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 def DigitalOutput.DigitalOutput.__init__(self)

This function is a constructor for the [DigitalOutput](#) class.

It creates the internal variables required to perform functions within the class. This function does not initialize any hardware.

Parameters

self	This object pointer
----------------------	---------------------

3.3.2.2 def DigitalOutput.DigitalOutput.__del__(self)

This is the destructor for the [DigitalOutput](#) Class.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3 Member Function Documentation

3.3.3.1 def DigitalOutput.DigitalOutput.close (*self*)

This function will close connection to the digital output device and channels.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.2 def DigitalOutput.DigitalOutput.createTestBuffer (*self*)

This function returns a random 1D numpy array of samples for writing the buffer of digital output channels.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.3 def DigitalOutput.DigitalOutput.getClkSource (*self*)

This function returns the sample clock source configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.4 def DigitalOutput.DigitalOutput.getNumChannels (*self*)

This function returns the number of digital channels configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.5 def DigitalOutput.DigitalOutput.getNumLines (*self*)

This function returns the number of digital lines configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.6 def DigitalOutput.DigitalOutput.getSampleRate (*self*)

This function returns the sample rate configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.7 `def DigitalOutput.DigitalOutput.getSamplesPerChannel (self)`

This function returns the samples per channel configured in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.8 `def DigitalOutput.DigitalOutput.init (self, physicalChannel)`

Initialize the digital outputs based on the object's configuration.

Parameters

<i>self</i>	The object pointer.
<i>physicalChannel</i>	A string representing the device and digital output channels. Example value: "PXI1Slot3/ao0-:7"

3.3.3.9 `def DigitalOutput.DigitalOutput.setClkSource (self, value)`

This function sets the sample clock source in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The value to set the clock source.

3.3.3.10 `def DigitalOutput.DigitalOutput.setSampleRate (self, value)`

This function sets the sample rate in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The value to set the sample rate.

3.3.3.11 `def DigitalOutput.DigitalOutput.setSamplesPerChannel (self, value)`

This function sets the samples per channel in the DAQmx Task.

Parameters

<i>self</i>	The object pointer.
<i>value</i>	The value to set the samples per channel.

3.3.3.12 `def DigitalOutput.DigitalOutput.start (self)`

This function starts the digital output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.13 def DigitalOutput.DigitalOutput.stop (*self*)

This function stops the digital output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.14 def DigitalOutput.DigitalOutput.waitUntilDone (*self*)

This functions waits for the digital output generation to complete.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.3.3.15 def DigitalOutput.DigitalOutput.writeToBuffer (*self*, *data*)

This function writes the specified values into the buffer.

Parameters

<i>self</i>	The object pointer.
<i>data</i>	This is a 1D 8-bit unsigned integer array that contains samples for each digital channel. Channels are non-interleaved (channel1 n-samples then channel2 n-samples).

3.3.4 Member Data Documentation

3.3.4.1 DigitalOutput.DigitalOutput.initialized

This is a boolean that is true when the DAQmx task has been initialized.

3.3.4.2 DigitalOutput.DigitalOutput.loops

The number of time to iterate over a Finite number of samples.

This value is only useful in the "Finite" mode. It is the number of times that a sequence of digital samples will be looped. The default is always 1.

3.3.4.3 DigitalOutput.DigitalOutput.mode

This is the mode of operation for the digital outputs.

There are currently three modes available. Static mode is where one static digital sample is set with no need for a sample clock. Finite mode is where a finite number of digital samples will be set at a sample clock rate. Continuous mode is where a sequence of voltages are generated at a sample rate and then repeated until the [stop\(\)](#) method is called.

3.3.4.4 DigitalOutput.DigitalOutput.status

This is the status of the DAQmx task.

A value greater than 0 means that an error has occurred. When the status is greater than 0 an error should be reported by the class.

3.3.4.5 DigitalOutput.DigitalOutput.taskRef

The DAQmx task reference.

3.3.5 Property Documentation

3.3.5.1 DigitalOutput.DigitalOutput.clkSource [static]

Initial value:

```
1 property(getClkSource, setClkSource, _delClkSource,
2          ""The clock source for the digital outputsample clock."")
```

This is the sample clock source terminal.

It can be set to an internal clock or external clock such as a PFI line i.e. "/PXI1Slot3/PFI15."

3.3.5.2 DigitalOutput.DigitalOutput.sampleRate [static]

Initial value:

```
1 property(getSampleRate, setSampleRate, _delSampleRate, doc=
2          ""The sample rate of the digital output."")
```

This is the sample rate of the digital output.

3.3.5.3 DigitalOutput.DigitalOutput.samplesPerChannel [static]

Initial value:

```
1 property(getSamplesPerChannel, setSamplesPerChannel,
2          _delSamplesPerChannel,
3          ""The samples per channel of the digital output."")
```

This is the number of samples per channel that will be generated in Finite mode.

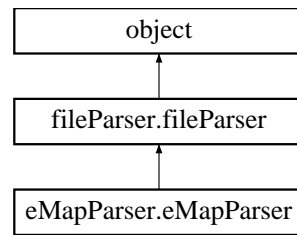
The documentation for this class was generated from the following file:

- DigitalOutput.py

3.4 eMapParser.eMapParser Class Reference

This class will parse the electrode map file.

Inheritance diagram for eMapParser.eMapParser:



Public Member Functions

- def [read](#)

This function will read all of the data from the eMap file and return a list of electrodes, ao numbers, and dsub numbers.

Additional Inherited Members

3.4.1 Detailed Description

This class will parse the electrode mape file.

3.4.2 Member Function Documentation

3.4.2.1 def eMapParser.eMapParser.read (self)

This function will read all of the data from the eMap file and return a list of electrodes, ao numbers, and dsub numbers.

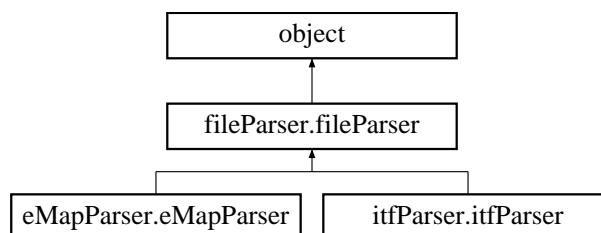
The documentation for this class was generated from the following file:

- eMapParser.py

3.5 fileParser.fileParser Class Reference

This class is the parent class of all file parsers.

Inheritance diagram for fileParser.fileParser:



Public Member Functions

- def [__init__](#)

This function is the constructor of the [fileParser](#) class.

- def [open](#)

This function opens the file in the filePath argument.

- def [getNumLines](#)

This function will return the number of lines within the file.

- `def close`

This function will close the file.

Public Attributes

- `fileObj`

The file object for the file.

- `comments`

The comments within the file as a list.

- `meta`

The meta data of the file as a list.

- `empty`

A boolean value that is true if the file is empty, and false otherwise.

Properties

- `totalLines`

This is the total number of lines of data within the itf file.

3.5.1 Detailed Description

This class is the parent class of all file parsers.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 `def fileParser.fileParser.__init__(self, fileObj=None, filePath=None)`

This function is the constructor of the `fileParser` class.

It creates class data and opens the file if the `filePath` argument is provided.

Parameters

<code>self</code>	The object pointer.
<code>fileObj</code>	The file object that gets returned by the file <code>open()</code> function.
<code>filePath</code>	A string that represents the location of the file.

3.5.3 Member Function Documentation

3.5.3.1 `def fileParser.fileParser.close (self)`

This function will close the file.

Parameters

<code>self</code>	The object pointer.
-------------------	---------------------

3.5.3.2 `def fileParser.fileParser.getNumLines (self)`

This function will return the number of lines within the file.

This function is equivalent to reading the totalLines variable within the class.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.5.3.3 def fileParser.fileParser.open (*self*, *filePath*)

This function opens the file in the filePath argument.

Parameters

<i>self</i>	The object pointer.
<i>filePath</i>	This is the path to the file.

3.5.4 Member Data Documentation

3.5.4.1 fileParser.fileParser.comments

The comments within the file as a list.

3.5.4.2 fileParser.fileParser.empty

A boolean value that is true if the file is empty, and false otherwise.

3.5.4.3 fileParser.fileParser.fileObj

The file object for the file.

3.5.4.4 fileParser.fileParser.meta

The meta data of the file as a list.

The dt variable is often within an itf file. The value of this variable will be a value within meta.

3.5.5 Property Documentation

3.5.5.1 fileParser.fileParser.totalLines [static]

Initial value:

```
1 property(getNumLines,
2          doc = """The total number of lines in the file.""")
```

This is the total number of lines of data within the itf file.

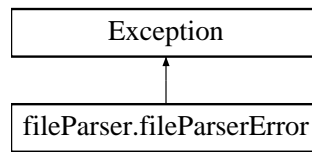
The documentation for this class was generated from the following file:

- fileParser.py

3.6 fileParser.fileParserError Class Reference

This is a class that creates a specific exception for the [fileParser](#) class.

Inheritance diagram for fileParser.fileParserError:



Public Member Functions

- `def __init__`
This is the constructor for the `fileParserError` class.
- `def __str__`
This function defines how the class operates when a the class is asked to represent itself as a string.

Public Attributes

- `msg`
The message to display to the user.

3.6.1 Detailed Description

This is a class that creates a specific exception for the `fileParser` class.

Having a specific exception help when trying to explain to the user why a certain function generated an error.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 `def fileParser.fileParserError.__init__(self, msg)`

This is the constructor for the `fileParserError` class.

This function creates the `msg` class attribute.

Parameters

<code>self</code>	The object pointer.
<code>msg</code>	The message to display to the user.

3.6.3 Member Function Documentation

3.6.3.1 `def fileParser.fileParserError.__str__(self)`

This function defines how the class operates when a the class is asked to represent itself as a string.

This function just returns the value of the `msg` class attribute.

3.6.4 Member Data Documentation

3.6.4.1 `fileParser.fileParserError.msg`

The message to display to the user.

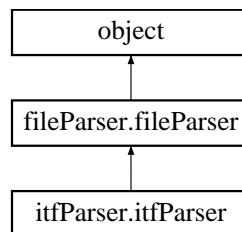
The documentation for this class was generated from the following file:

- fileParser.py

3.7 itfParser.itfParser Class Reference

This class will parse through an itf file.

Inheritance diagram for itfParser.itfParser:



Public Member Functions

- `def __init__`
This function is the constructor of the `itfParser` class.
- `def readline`
This function will read a line from the itf file and returns the data as a dictionary.
- `def eMapReadLine`
This function reads a line from the itf file and uses an electrode map file to sort the data by the electrode order within the file.
- `def readlines`
This function will read the number of lines specified by the `numLines` argument and return the data as a dictionary.
- `def eMapReadLines`
This function reads lines from the itf file and uses an electrode map file to sort the data by the electrode order within the file.
- `def read`
This function will read all of the lines within the file and returns the data as a dictionary.
- `def eMapRead`
This function reads all lines from the itf file and uses an electrode map file to sort the data by the electrode order within the file.
- `def appendline`
This function will append a line of data to the itf file.

Public Attributes

- `tableHeader`
This is the names of the columns for the table header of the itf file.
- `eMapFilePath`
This is the file path to the electrode map file.
- `comments`
The comments within the file as a list.
- `meta`
The meta data of the file as a dictionary.
- `empty`
A boolean file that is true if the file is empty, and false otherwise.

Additional Inherited Members

3.7.1 Detailed Description

This class will parse through an itf file.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 `def itfParser.itfParser.__init__(self, fileObj=None, filePath=None)`

This function is the constructor of the `itfParser` class.

It initializes class attributes.

Parameters

<i>self</i>	The object pointer.
<i>fileObj</i>	A file object pointer for a file that has already been opened.
<i>filePath</i>	A path to the itf file that hasn't already been opened.

3.7.3 Member Function Documentation

3.7.3.1 `def itfParser.itfParser.appendline (self, data)`

This function will append a line of data to the itf file.

Parameters

<i>self</i>	The object reference.
<i>data</i>	This will take multiple data types. Supported types are numpy 1D float64, a list, and a dictionary.

3.7.3.2 `def itfParser.itfParser.eMapRead (self, eMapFilePath=None)`

This function reads all lines from the itf file and uses an electrode map file to sort the data by the electrode order within the file.

This function returns a numpy float64 array, which is compatible with the array format accepted by the WaveformChassis class.

Parameters

<i>self</i>	The object pointer.
<i>eMapFilePath</i>	The file path to the electrode map.

3.7.3.3 `def itfParser.itfParser.eMapReadLine (self, lineNum=None, eMapFilePath=None)`

This function reads a line from the itf file and uses an electrode map file to sort the data by the electrode order within the file.

This function returns a numpy float64 array, which is compatible with the array format accepted by the WaveformChassis class.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

<i>lineNum</i>	If this argument is provided then the line number referred to by this argument is returned. Otherwise the next line is returned. (This argument is zero-based. Meaning that passing a value of zero will return the first line in the file.)
<i>eMapFilePath</i>	The file path to the electrode map.

3.7.3.4 def itfParser.itfParser.eMapReadLines (*self*, *numLines*, *eMapFilePath* = None)

This function reads lines from the itf file and uses an electrode map file to sort the data by the electrode order within the file.

This function returns a numpy float64 array, which is compatible with the array format accepted by the WaveformChassis class.

Parameters

<i>self</i>	The object pointer.
<i>numLines</i>	The number of lines to return from the itf file.
<i>eMapFilePath</i>	The file path to the electrode map.

3.7.3.5 def itfParser.itfParser.read (*self*)

This function will read all of the lines within the file and returns the data as a dictionary.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.7.3.6 def itfParser.itfParser.readline (*self*, *lineNum* = None)

This function will read a line from the itf file and returns the data as a dictionary.

Parameters

<i>self</i>	The object pointer.
<i>lineNum</i>	If this argument is provided then the line number referred to by this argument is returned. Otherwise the next line is returned. (This argument is zero-based. Meaning that passing a value of zero will return the first line in the file.)

3.7.3.7 def itfParser.itfParser.readlines (*self*, *numLines*)

This function will read the number of lines specified by the numLines argument and return the data as a dictionary.

Parameters

<i>self</i>	The object pointer.
<i>numLines</i>	The number of lines to pull from the file.

3.7.4 Member Data Documentation

3.7.4.1 itfParser.itfParser.comments

The comments within the file as a list.

3.7.4.2 itfParser.itfParser.eMapFilePath

This is the file path to the electrode map file.

If this variable is populated then this file path will be used when a [eMapReadLine\(\)](#) method is called.

3.7.4.3 itfParser.itfParser.empty

A boolean file that is true if the file is empty, and false otherwise.

3.7.4.4 itfParser.itfParser.meta

The meta data of the file as a dictionary.

3.7.4.5 itfParser.itfParser.tableHeader

This is the names of the columns for the table header of the itf file.

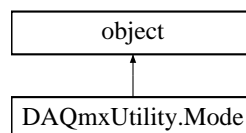
The documentation for this class was generated from the following file:

- itfParser.py

3.8 DAQmxUtility.Mode Class Reference

This class defines the mode variable which is used as an enumerated typedef.

Inheritance diagram for DAQmxUtility.Mode:



Static Public Attributes

- int [Finite](#) 0
The Finite mode is a mode where a finite number of samples are generated.
- int [Continuous](#) 1
The Continuous mode is a mode where a set of samples are continuously repeated.
- int [Static](#) 2
The Static mode is a mode where only one sample is generated.

3.8.1 Detailed Description

This class defines the mode variable which is used as an enumerated typedef.

3.8.2 Member Data Documentation

3.8.2.1 int DAQmxUtility.Mode.Continuous 1 [static]

The Continuous mode is a mode where a set of samples are continuously repeated.

They samples are repeated until the stop() method is called. Default value: 1.

3.8.2.2 int DAQmxUtility.Mode.Finite 0 [static]

The Finite mode is a mode where a finite number of samples are generated.
Default value: 0.

3.8.2.3 int DAQmxUtility.Mode.Static 2 [static]

The Static mode is a mode where only one sample is generated.
Default Value: 2.

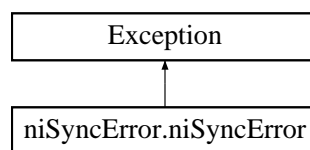
The documentation for this class was generated from the following file:

- DAQmxUtility.py

3.9 niSyncError.niSyncError Class Reference

This is a class to handle errors generated by the niSync python module.

Inheritance diagram for niSyncError.niSyncError:



Public Member Functions

- `def __init__`
This is the constructor of the [niSyncError](#) class.
- `def __str__`
This function returns a string containing the code and message describing the error that occurred.

Public Attributes

- `code`
This is the status code returned by the NI-Sync drivers.
- `msg`
This is the message returned by the NI-Sync drivers.

3.9.1 Detailed Description

This is a class to handle errors generated by the niSync python module.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 `def niSyncError.niSyncError.__init__(self, code, msg)`

This is the constructor of the [niSyncError](#) class.

3.9.3 Member Function Documentation

3.9.3.1 `def niSyncError.niSyncError.__str__(self)`

This function returns a string containing the code and message describing the error that occurred.

This string will be displayed to the stderr output when a NI-Sync error occurs.

3.9.4 Member Data Documentation

3.9.4.1 `niSyncError.niSyncError.code`

This is the status code returned by the NI-Sync drivers.

The code value is useful, because error codes can be searched on the National Instruments website (www.ni.com).

3.9.4.2 `niSyncError.niSyncError.msg`

This is the message returned by the NI-Sync drivers.

The message gives a more detailed explanation as to why an error occurred.

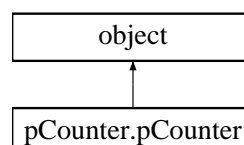
The documentation for this class was generated from the following file:

- `niSyncError.py`

3.10 `pCounter.pCounter` Class Reference

This class will count edges of a digital signal at a sample clock rate specified, and has capability of being triggered to start.

Inheritance diagram for `pCounter.pCounter`:



Public Member Functions

- `def __init__`
This function is the constructor for the `pCounter` class.
- `def init`
This function initializes the `pCounter` class and opens a reference to the DAQmx device(s).
- `def initFromFile`
This function initializes the `pCounter` class using the chassis config file and opens a reference to the DAQmx device(s).
- `def start`
This function starts the measurement.
- `def stop`
This function stops the measurement.
- `def read`
This function returns an array of the edge counts with an array size equal to the number of samples.

- def `measure`
This function performs the `start()`, `read()`, and `stop()` methods in one function call.
- def `close`
This function closes the references to the DAQmx devices.
- def `__del__`
This function is the destructor for the `pCounter` class.

Public Attributes

- `edgeCounter`
The string that identifies the DAQmx device and counter for the counter that is used to count edges.
- `clockCounter`
The string that identifies the DAQmx device and counter for the counter that is used to create the sample clock.
- `enableStartTrigger`
A boolean that enables the start trigger.
- `edgeCntrTerm`
A string that identifies the DAQmx digital line that will be used as an input to the edge counter.
- `triggerSource`
A string that identifies the DAQmx digital line that will be used as the start trigger.
- `clockSourceTerm`
A string that identifies the DAQmx digital line that will output the sample clock.
- `edgeCntrTask`
The task reference for the edge counter.
- `clockCntrTask`
The task reference for the sample clock counter.
- `timeout`
This is the time to wait for a start trigger.

Properties

- `samples` property(`_getSamples`, `_setSamples`)
This is the number of samples to take.
- `sampleRate` property(`_getSampleRate`, `_setSampleRate`)
This is the sample rate to use when counting edges.
- `binTime` property(`_getBinTime`, `_setBinTime`)
This is the time in milliseconds to take a single sample.
- `acqTime` property(`_getAcqTime`, `_setAcqTime`)
This is the time in milliseconds for a full acquisition period.

3.10.1 Detailed Description

This class will count edges of a digital signal at a sample clock rate specified, and has capability of being triggered to start.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 `def pCounter.pCounter.__init__ (self)`

This function is the constructor for the `pCounter` class.

It creates internal variables required to perform functions within the class. This function does not initialize any hardware.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.2.2 `def pCounter.pCounter.__del__ (self)`

This function is the destructor for the `pCounter` class.

It deletes internal variables and closes the references to the DAQmx devices if they are not already closed.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.3 Member Function Documentation

3.10.3.1 `def pCounter.pCounter.close (self)`

This function closes the references to the DAQmx devices.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.3.2 `def pCounter.pCounter.init (self, clockCounter=None, edgeCounter=None, acqTime=None, binTime=None, samples=None, sampleRate=None)`

This function initializes the `pCounter` class and opens a reference to the DAQmx device(s).

Parameters

<i>self</i>	The object pointer.
<i>clockCounter</i>	The string that identifies the DAQmx device and counter for the counter that is used to create the sample clock.
<i>edgeCounter</i>	The string that identifies the DAQmx device and counter for the counter that is used to count edges.
<i>acqTime</i>	This is the time in milliseconds for a full acquisition period.
<i>binTime</i>	This is the time in milliseconds to take a single sample.

3.10.3.3 `def pCounter.pCounter.initFromFile (self, filepath)`

This function initializes the `pCounter` class using the chassis config file and opens a reference to the DAQmx device(s).

Parameters

<i>self</i>	The object reference.
<i>filepath</i>	The path to the chassis config file.

3.10.3.4 def pCounter.pCounter.measure (*self*)

This function performs the [start\(\)](#), [read\(\)](#), and [stop\(\)](#) methods in one function call.

This is useful for when the results of the [read\(\)](#) method can be retrieved immediately after a [start\(\)](#)

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.3.5 def pCounter.pCounter.read (*self*)

This function returns an array of the edge counts with an array size equal to the number of samples.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.3.6 def pCounter.pCounter.start (*self*)

This function starts the measurement.

If the start trigger is enabled, then a the [pCounter](#) waits for that digital trigger. Otherwise the measurement takes place immediately.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.3.7 def pCounter.pCounter.stop (*self*)

This function stops the measurement.

It needs to be called everytime the [start\(\)](#) method is called.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.10.4 Member Data Documentation

3.10.4.1 pCounter.pCounter.clockCntrTask

The task reference for the sample clock counter.

3.10.4.2 pCounter.pCounter.clockCounter

The string that identifies the DAQmx device and counter for the counter that is used to create the sample clock.

Example: /PXI1Slot5/ctr0

3.10.4.3 `pCounter.pCounter.clockSourceTerm`

A string that identifies the DAQmx digital line that will output the sample clock.

Default: PFI12

3.10.4.4 `pCounter.pCounter.edgeCntrTask`

The task reference for the edge counter.

3.10.4.5 `pCounter.pCounter.edgeCntrTerm`

A string that identifies the DAQmx digital line that will be used as an input to the edge counter.

Default: PFI0

3.10.4.6 `pCounter.pCounter.edgeCounter`

The string that identifies the DAQmx device and counter for the counter that is used to count edges.

Example: PXISlot5/ctr0

3.10.4.7 `pCounter.pCounter.enableStartTrigger`

A boolean that enables the start trigger.

The default is false, which disables the start trigger. The measurement will immediately start when the [start\(\)](#) method is called. A true value will make the measurement start when a digital trigger is received on the line specified by the `triggerSource` variable.

3.10.4.8 `pCounter.pCounter.timeout`

This is the time to wait for a start trigger.

If the timeout passes, then an error is generated. Ignore this variable if the start trigger is disabled.

3.10.4.9 `pCounter.pCounter.triggerSource`

A string that identifies the DAQmx digital line that will be used as the start trigger.

Default: PFI1

3.10.5 Property Documentation

3.10.5.1 `pCounter.pCounter.acqTime` `property(_getAcqTime, _setAcqTime)` `[static]`

This is the time in milliseconds for a full acquisition period.

3.10.5.2 `pCounter.pCounter.binTime` `property(_getBinTime, _setBinTime)` `[static]`

This is the time in milliseconds to take a single sample.

3.10.5.3 pCounter.pCounter.sampleRate property([_getSampleRate](#), [_setSampleRate](#)) [[static](#)]

This is the sample rate to use when counting edges.

3.10.5.4 pCounter.pCounter.samples property([_getSamples](#), [_setSamples](#)) [[static](#)]

This is the number of samples to take.

It is the size of the data array returned by the [read\(\)](#) method.

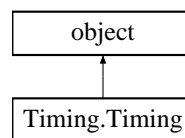
The documentation for this class was generated from the following file:

- [pCounter.py](#)

3.11 Timing.Timing Class Reference

This class will generate perform the timing and synchronization required to synchronize all cards on the PXI back-plane.

Inheritance diagram for Timing.Timing:



Public Member Functions

- [def __init__](#)
This function is a constructor for the [Timing](#) class.
- [def init](#)
This function connects to the niSync device and starts to setup clock and trigger connections.
- [def sendSoftwareTrigger](#)
This function will send a trigger to all devices when it is called.
- [def close](#)
This function will close the connection to the niSync device.
- [def __del__](#)
This is the destructor for the [Timing](#) class.

Public Attributes

- [session](#)
This is the reference to the NI-Sync session.
- [resourceName](#)
This the name of the NI-Sync resource that refers to the synchronization PXI card.
- [status](#)
This is the status of the NI-Sync session.
- [initialized](#)
This is a boolean that is true when the NI-Sync session has been initialized.
- [sampleRate](#)
This is the sample rate of the sample clock that gets distributed to all of the cards on a PXI Chassis.

- [divisor](#)

The dds is used to generate the sample clock.

- [ddsFreq](#)

This is the DDS frequency that is calculated from the sample rate and the divisor.

- [pxiStarSlots](#)

This is the number of pxi star slots available on the chassis.

3.11.1 Detailed Description

This class will generate perform the timing and synchronization required to synchronize all cards on the PXI back-plane.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 `def Timing.Timing.__init__(self)`

This function is a constructor for the [Timing](#) class.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.11.2.2 `def Timing.Timing.__del__(self)`

This is the destructor for the [Timing](#) class.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.11.3 Member Function Documentation

3.11.3.1 `def Timing.Timing.close (self)`

This function will close the connection to the niSync device.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.11.3.2 `def Timing.Timing.init (self, deviceName)`

This function connects to the niSync device and starts to setup clock and trigger connections.

Parameters

<i>self</i>	The object pointer.
<i>deviceName</i>	The name of the niSync device. This name can be found in Measurement and Automation Explorer. Example value: "PXI1Slot14"

3.11.3.3 def Timing.Timing.sendSoftwareTrigger (self)

This function will send a trigger to all devices when it is called.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.11.4 Member Data Documentation

3.11.4.1 Timing.Timing.ddsFreq

This is the DDS frequency that is calculated from the sample rate and the divisor.

The DDS frequency is the sample rate divided by the divisor.

3.11.4.2 Timing.Timing.divisor

The dds is used to generate the sample clock.

The lower the value of the divisor the more inaccurate the sample clock. This value is defaulted to its maximum of 32.

3.11.4.3 Timing.Timing.initialized

This is a boolean that is true when the NI-Sync session has been initialized.

3.11.4.4 Timing.Timing.pxiStarSlots

This is the number of pxi star slots available on the chassis.

In order for the sample clock to be distributed to all cards, this value must be equal to or larger than the number of potential cards on the Chassis.

3.11.4.5 Timing.Timing.resourceName

This the name of the NI-Sync resource that refers to the synchronization PXI card.

3.11.4.6 Timing.Timing.sampleRate

This is the sample rate of the sample clock that gets distributed to all of the cards on a PXI Chassis.

3.11.4.7 Timing.Timing.session

This is the reference to the NI-Sync session.

3.11.4.8 Timing.Timing.status

This is the status of the NI-Sync session.

A value greater than 0 means that an error has occurred. When the status is greater than 0 an error should be reported by the class.

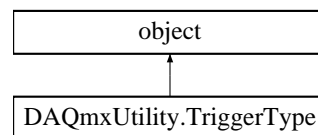
The documentation for this class was generated from the following file:

- Timing.py

3.12 DAQmxUtility.TriggerType Class Reference

This class defines the trigger type variable which is used as an enumerated typedef.

Inheritance diagram for DAQmxUtility.TriggerType:



Static Public Attributes

- int [Software](#) 0
The Software trigger type is used when a single software command starts the signal generation.
- int [Hardware](#) 1
The Hardware trigger types is used when a start trigger is utilized to start the signal generation.

3.12.1 Detailed Description

This class defines the trigger type variable which is used as an enumerated typedef.

3.12.2 Member Data Documentation

3.12.2.1 int DAQmxUtility.TriggerType.Hardware 1 [static]

The Hardware trigger types is used when a start trigger is utilized to start the signal generation.

A software command may still be used to start the generation. However, that command will set the generator to wait for a trigger such that all signal are synchronized. Default value: 1

3.12.2.2 int DAQmxUtility.TriggerType.Software 0 [static]

The Software trigger type is used when a single software command starts the signal generation.

Default value: 0.

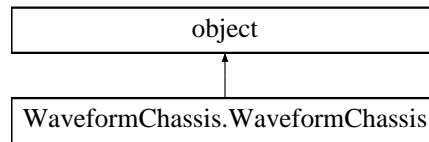
The documentation for this class was generated from the following file:

- DAQmxUtility.py

3.13 WaveformChassis.WaveformChassis Class Reference

This class contains a list of WaveformGenerator objects and a Timing object.

Inheritance diagram for WaveformChassis.WaveformChassis:



Public Member Functions

- `def __init__`
This function is a constructor for the [WaveformChassis](#) class.
- `def init`
Initializes the waveform chassis based on the object's configuration.
- `def initFromFile`
This function will initialize the waveform chassis based on the configuration file specified by the `filePath` parameter.
- `def getNumAoChannels`
This function returns the total number of AO channels configured.
- `def getNumDoChannels`
This function returns the total number of DO channels configured.
- `def createAoSineBuffer`
This function returns a 1D numpy array of sine waves - one for each channel configured by the `init` function.
- `def createDoTestBuffer`
This function returns a 1d array of random U8s for writing to the buffer of digital output channels.
- `def writeAoBuffer`
This function will write data into the buffer of the analog outputs that are a part of the [WaveformChassis](#) class.
- `def writeDoBuffer`
This function will write data into the buffer of the digital outputs that are a part of the [WaveformChassis](#).
- `def start`
This function starts the analog and digital output generation.
- `def waitUntilDone`
This functions waits for the analog and digital output generation to complete.
- `def stop`
This function stops the analog and digital output generation.
- `def writeStartStop`
This function will perform the `write`, `start`, `waitUntilDone`, and `stop` functions wrapped in one function.
- `def close`
This function will close connection to the [WaveformChassis](#).

Public Attributes

- `timing`
This is a [Timing](#) object created using the [Timing](#) class.
- `useTiming`
This is a boolean value that will be true if the [WaveformChassis](#) is configured to use [Timing](#).
- `gens`
This is a list of [WaveformGenerator](#) objects created using the [WaveformGenerator](#) class.
- `sampleRate`
This is the sample rate of all the Analog and Digital outputs.
- `samplesPerChannel`
This is the number of channels configured for the [WaveformChassis](#).
- `mode`

This is the mode of operation for all generators.

- [loops](#)

The number of times to iterate over a Finite number of samples.

- [triggerType](#)

The trigger type for the analog outputs.

- [clkSource](#)

This is the sample clock source terminal for every WaveformGenerator configured in the [WaveformChassis](#).

- [startTriggerSource](#)

This is the start trigger source terminal for every WaveformGenerator configured in the [WaveformChassis](#).

3.13.1 Detailed Description

This class contains a list of WaveformGenerator objects and a Timing object.

It is intended to represent a chassis with a NiSync Card and a number of PXI-6733s.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 `def WaveformChassis.WaveformChassis.__init__(self)`

This function is a constructor for the [WaveformChassis](#) class.

It creates the internal variables required to perform functions within the class. This function does not initialize any hardware.

3.13.3 Member Function Documentation

3.13.3.1 `def WaveformChassis.WaveformChassis.close(self)`

This function will close connection to the [WaveformChassis](#).

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.2 `def WaveformChassis.WaveformChassis.createAoSineBuffer(self)`

This function returns a 1D numpy array of sine waves - one for each channel configured by the init function.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.3 `def WaveformChassis.WaveformChassis.createDoTestBuffer(self)`

This function returns a 1d array of random U8s for writing to the buffer of digital output channels.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.4 `def WaveformChassis.WaveformChassis.getNumAoChannels (self)`

This function returns the total number of AO channels configured.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.5 `def WaveformChassis.WaveformChassis.getNumDoChannels (self)`

This function returns the total number of AO channels configured.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.6 `def WaveformChassis.WaveformChassis.init (self, aoDevsAndChnls, doDevsAndChnls = None, syncDevice = None)`

Initializes the waveform chassis based on the object's configuration.

Parameters

<i>self</i>	The object pointer.
<i>aoDevsAndChnls</i>	This is a list of strings representing the devices and analog output channels.
<i>doDevsAndChnls</i>	This is a list of strings representing the divices and digital output channels.
<i>syncDevice</i>	This is the device name for the NI Sync card.

3.13.3.7 `def WaveformChassis.WaveformChassis.initFromFile (self, filePath)`

This function will initialize the waveform chassis based on the configuration file specified by the filePath parameter.

Parameters

<i>self</i>	The object pointer
<i>filePath</i>	The file path to the configuration file.

3.13.3.8 `def WaveformChassis.WaveformChassis.start (self)`

This function starts the analog and digital output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.9 `def WaveformChassis.WaveformChassis.stop (self)`

This function stops the analog and digital output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.10 `def WaveformChassis.WaveformChassis.waitUntilDone (self)`

This functions waits for the analog and digital output generation to complete.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.3.11 `def WaveformChassis.WaveformChassis.writeAoBuffer (self, data)`

This function will write data into the buffer of the analog outputs that are a part of the [WaveformChassis](#) class.

Parameters

<i>self</i>	The object pointer
<i>data</i>	This is a 1D 8-bit unsigned integer array that contains data for each digital channel. Channels are non-interleaved (channel1 n-samples then channel2 n-samples).

3.13.3.12 `def WaveformChassis.WaveformChassis.writeDoBuffer (self, data)`

This function will write data into the buffer of the digital outputs that are a part of the [WaveformChassis](#).

Parameters

<i>self</i>	The object pointer.
<i>data</i>	This is a 1D 8-bit unsigned integer array that contains data for each digital channel. Channels are non-interleaved (channel1 n-samples then channel2 n-samples).

3.13.3.13 `def WaveformChassis.WaveformChassis.writeStartStop (self, aoData, doData=None)`

This function will perform the write, start, waitUntilDone, and stop functions wrapped in one function.

This function is not complete and may not exist in a later release!!!

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.13.4 Member Data Documentation

3.13.4.1 `WaveformChassis.WaveformChassis.clkSource`

This is the sample clock source terminal for every WaveformGenerator configured in the [WaveformChassis](#).

It is defaulted to "PXI_Star."

3.13.4.2 `WaveformChassis.WaveformChassis.gens`

This is a list of WaveformGenerator objects created using the WaveformGenerator class.

It is a list of PXI6733 cards configured for the [WaveformChassis](#).

3.13.4.3 WaveformChassis.WaveformChassis.loops

The number of times to iterate over a Finite number of samples.

This value is only useful in the "Finite" mode. It is the number of times that a sequence of voltages will be looped. The default is always 1.

3.13.4.4 WaveformChassis.WaveformChassis.mode

This is the mode of operation for all generators.

There are currently three modes available. Static mode is where one static voltage is set with no need for a sample clock. Finite mode is where a finite number of voltages will be set at a sample clock rate. Continuous mode is where a sequence of voltages are generated at a sample rate and then repeated until the [stop\(\)](#) method is called.

3.13.4.5 WaveformChassis.WaveformChassis.sampleRate

This is the sample rate of all the Analog and Digital outputs.

The sample rate must be the same for all devices configured in the [WaveformChassis](#).

3.13.4.6 WaveformChassis.WaveformChassis.samplesPerChannel

This is the number of channels configured for the [WaveformChassis](#).

The samples per channel must be the same for all devices configured in the [WaveformChassis](#).

3.13.4.7 WaveformChassis.WaveformChassis.startTriggerSource

This is the start trigger source terminal for every WaveformGenerator configured in the [WaveformChassis](#).

It is defaulted to "PXI_Trig1"

3.13.4.8 WaveformChassis.WaveformChassis.timing

This is a Timing object created using the Timing class.

It is used to synchronize all of the PXI cards.

3.13.4.9 WaveformChassis.WaveformChassis.triggerType

The trigger type for the analog outputs.

There are currently two trigger types - "Software" and "Hardware." The "Software" mode means that analog output channels are not synchronized. While "Hardware" means that analog output channels are synchronized to a start trigger. The startTriggerSouce attribute must be configured appropriately.

3.13.4.10 WaveformChassis.WaveformChassis.useTiming

This is a boolean value that will be true if the [WaveformChassis](#) is configured to use Timing.

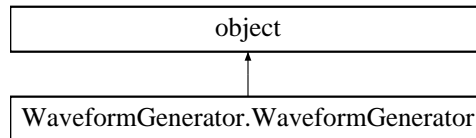
The documentation for this class was generated from the following file:

- WaveformChassis.py

3.14 WaveformGenerator.WaveformGenerator Class Reference

This class contains an AnalogOutput and DigitalOutput object.

Inheritance diagram for WaveformGenerator.WaveformGenerator:



Public Member Functions

- `def __init__`
This function is a constructor for the [WaveformGenerator](#) class.
- `def init`
Initialize the analog and digital outputs based on the object's configuration.
- `def writeAoBuffer`
This function writes the specified values into the buffer.
- `def writeDoBuffer`
This function writes the specified values into the buffer.
- `def start`
This function starts the analog and digital output generation.
- `def waitUntilDone`
This functions waits for the analog and digital output generation to complete.
- `def stop`
This function stops the analog and digital output generation.
- `def writeStartStop`
- `def close`
This function will close connection to the analog and digital ouput device and channels.

Public Attributes

- `ao`
This is an AnalogOutput object created using the AnalogOutput class.
- `do`
This is a DigitalOutput object created using the DigitaOutput class.
- `sampleRate`
This is the sample rate of the waveform generator.
- `samplesPerChannel`
This is the number of channels configured in the waveform generator.
- `clkSource`
This is the sample clock source terminal.
- `startTriggerSource`
This is the start trigger source terminal.
- `mode`
There are currently three modes available.
- `loops`
This value is only useful in the "Finite" mode.
- `triggerType`

There are currently two trigger types - "Software" and "Hardware." The "Software" mode means that analog output channels are not synchronized.

- [useAo](#)

This is a boolean value that will be true if the [WaveformGenerator](#) is configured to utilize analog outputs.

- [useDo](#)

This is a boolean value that will be true if the [WaveformGenerator](#) is configured to utilize digital outputs.

3.14.1 Detailed Description

This class contains an AnalogOutput and DigitalOutput object.

It is intended to represent a single PXI-6733 in software.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 `def WaveformGenerator.WaveformGenerator.__init__(self)`

This function is a constructor for the [WaveformGenerator](#) class.

It creates the internal variables required to perform functions within the class. This function does not initialize any hardware.

3.14.3 Member Function Documentation

3.14.3.1 `def WaveformGenerator.WaveformGenerator.close(self)`

This function will close connection to the analog and digital output device and channels.

3.14.3.2 `def WaveformGenerator.WaveformGenerator.init(self, aoChannels, doChannels = ' ')`

Initialize the analog and digital outputs based on the object's configuration.

Parameters

<i>self</i>	The object pointer.
<i>aoChannels</i>	This is a string representing the device and analog output channels. Example value: "PXI1-Slot3/ao0:7"
<i>doChannels</i>	This is a string representing the device and digital output channels. Example value: "PXI1-Slot3/port0/line0:7"

3.14.3.3 `def WaveformGenerator.WaveformGenerator.start(self)`

This function starts the analog and digital output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.14.3.4 `def WaveformGenerator.WaveformGenerator.stop(self)`

This function stops the analog and digital output generation.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

3.14.3.5 `def WaveformGenerator.WaveformGenerator.waitUntilDone (self)`

This functions waits for the analog and digital output generation to complete.

3.14.3.6 `def WaveformGenerator.WaveformGenerator.writeAoBuffer (self, data)`

This function writes the specified values into the buffer.

Parameters

<i>self</i>	The object pointer.
<i>data</i>	This is a 1D 64-bit float numpy array that contains data for each analog output. Channels are non-interleaved (channel1 n-samples then channel2 n-samples).

3.14.3.7 `def WaveformGenerator.WaveformGenerator.writeDoBuffer (self, data)`

This function writes the specified values into the buffer.

Parameters

<i>self</i>	The object pointer.
<i>data</i>	This is a 1D 8-bit unsigned integer array that contains data for each digital line. Lines are non-interleaved (line1 n-samples then line2 n-samples).

3.14.4 Member Data Documentation

3.14.4.1 `WaveformGenerator.WaveformGenerator.ao`

This is an AnalogOutput object created using the AnalogOutput class.

3.14.4.2 `WaveformGenerator.WaveformGenerator.clkSource`

This is the sample clock source terminal.

It can be set to an internal clock or external clock such as a PFI line i.e. "/PXI1Slot3/PFI15."

3.14.4.3 `WaveformGenerator.WaveformGenerator.do`

This is a DigitalOutput object created using the DigitalOutput class.

3.14.4.4 `WaveformGenerator.WaveformGenerator.loops`

This value is only useful in the "Finite" mode.

It is the number of times that a sequence of voltages will be looped. The default is always 1.

3.14.4.5 `WaveformGenerator.WaveformGenerator.mode`

There are currently three modes available.

Static mode is where one static voltage is set with no need for a sample clock. Finite mode is where a finite number of voltages will be set at a sample clock rate. Continuous mode is where a sequence of voltages are generated at a sample rate and then repeated until the [stop\(\)](#) method is called.

3.14.4.6 WaveformGenerator.WaveformGenerator.sampleRate

This is the sample rate of the waveform generator.

3.14.4.7 WaveformGenerator.WaveformGenerator.samplesPerChannel

This is the number of channels configured in the waveform generator.

3.14.4.8 WaveformGenerator.WaveformGenerator.startTriggerSource

This is the start trigger source terminal.

The software ignores this value when the triggerType is set to "Software". Otherwise when the triggerType is "-Hardware," this terminal is used to start analog generation. Example Value: "/PXI1Slot3/PFI0"

3.14.4.9 WaveformGenerator.WaveformGenerator.triggerType

There are currently two trigger types - "Software" and "Hardware." The "Software" mode means that analog output channels are not synchronized.

While "Hardware" means that analog output channels are synchronized to a start trigger. The startTriggerSource attribute must be configured appropriately.

3.14.4.10 WaveformGenerator.WaveformGenerator.useAo

This is a boolean value that will be true if the [WaveformGenerator](#) is configured to utilize analog outputs.

3.14.4.11 WaveformGenerator.WaveformGenerator.useDo

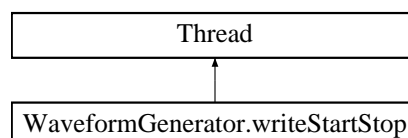
This is a boolean value that will be true if the [WaveformGenerator](#) is configured to utilize digital outputs.

The documentation for this class was generated from the following file:

- WaveformGenerator.py

3.15 WaveformGenerator.writeStartStop Class Reference

Inheritance diagram for WaveformGenerator.writeStartStop:



Public Member Functions

- def `__init__`
- def `run`

Public Attributes

- **wfrmGen**
- **aoBuffer**
- **doBuffer**

The documentation for this class was generated from the following file:

- WaveformGenerator.py

Index

- `__del__`
 - `AnalogOutput::AnalogOutput`, 7
 - `DigitalOutput::DigitalOutput`, 14
 - `pCounter::pCounter`, 30
 - `Timing::Timing`, 34
 - `__init__`
 - `AnalogOutput::AnalogOutput`, 7
 - `DigitalOutput::DigitalOutput`, 14
 - `fileParser::fileParser`, 20
 - `fileParser::fileParserError`, 22
 - `itfParser::itfParser`, 24
 - `niSyncError::niSyncError`, 27
 - `pCounter::pCounter`, 30
 - `Timing::Timing`, 34
 - `WaveformChassis::WaveformChassis`, 38
 - `WaveformGenerator::WaveformGenerator`, 43
 - `__str__`
 - `fileParser::fileParserError`, 22
 - `niSyncError::niSyncError`, 28
- `acqTime`
 - `pCounter::pCounter`, 32
- `AnalogOutput.AnalogOutput`, 5
- `AnalogOutput::AnalogOutput`
 - `__del__`, 7
 - `__init__`, 7
 - `clkSource`, 11
 - `close`, 7
 - `createSineTestBuffer`, 7
 - `createTestBuffer`, 7
 - `estAcqTime`, 10
 - `getClkSource`, 7
 - `getNumChannels`, 8
 - `getSampleRate`, 8
 - `getSamplesPerChannel`, 8
 - `getStartTriggerSource`, 8
 - `init`, 8
 - `initialized`, 10
 - `loops`, 10
 - `mode`, 10
 - `numChannels`, 11
 - `sampleRate`, 11
 - `samplesPerChannel`, 11
 - `setClkSource`, 8
 - `setSampleRate`, 9
 - `setSamplesPerChannel`, 9
 - `setStartTriggerSource`, 9
 - `start`, 9
 - `startTriggerSource`, 11
 - `startTriggerSyncCard`, 10
 - `status`, 10
 - `stop`, 9
 - `taskRef`, 10
 - `triggerType`, 11
 - `waitUntilDone`, 9
 - `writeToBuffer`, 10
- `ao`
 - `WaveformGenerator::WaveformGenerator`, 44
- `appendline`
 - `itfParser::itfParser`, 24
- `binTime`
 - `pCounter::pCounter`, 32
- `chassisConfigParser.chassisConfigParser`, 11
- `chassisConfigParser::chassisConfigParser`
 - `createDefaultFile`, 12
 - `read`, 12
 - `write`, 12
- `clkSource`
 - `AnalogOutput::AnalogOutput`, 11
 - `DigitalOutput::DigitalOutput`, 18
 - `WaveformChassis::WaveformChassis`, 40
 - `WaveformGenerator::WaveformGenerator`, 44
- `clockCntrTask`
 - `pCounter::pCounter`, 31
- `clockCounter`
 - `pCounter::pCounter`, 31
- `clockSourceTerm`
 - `pCounter::pCounter`, 31
- `close`
 - `AnalogOutput::AnalogOutput`, 7
 - `DigitalOutput::DigitalOutput`, 15
 - `fileParser::fileParser`, 20
 - `pCounter::pCounter`, 30
 - `Timing::Timing`, 34
 - `WaveformChassis::WaveformChassis`, 38
 - `WaveformGenerator::WaveformGenerator`, 43
- `code`
 - `niSyncError::niSyncError`, 28
- `comments`
 - `fileParser::fileParser`, 21
 - `itfParser::itfParser`, 25
- `Continuous`
 - `DAQmxUtility::Mode`, 26
- `createAoSineBuffer`
 - `WaveformChassis::WaveformChassis`, 38
- `createDefaultFile`
 - `chassisConfigParser::chassisConfigParser`, 12
- `createDoTestBuffer`

- WaveformChassis::WaveformChassis, 38
- createSineTestBuffer
 - AnalogOutput::AnalogOutput, 7
- createTestBuffer
 - AnalogOutput::AnalogOutput, 7
 - DigitalOutput::DigitalOutput, 15
- DAQmxUtility.Mode, 26
- DAQmxUtility.TriggerType, 36
- DAQmxUtility::Mode
 - Continuous, 26
 - Finite, 27
 - Static, 27
- DAQmxUtility::TriggerType
 - Hardware, 36
 - Software, 36
- ddsFreq
 - Timing::Timing, 35
- DigitalOutput.DigitalOutput, 13
- DigitalOutput::DigitalOutput
 - __del__, 14
 - __init__, 14
 - clkSource, 18
 - close, 15
 - createTestBuffer, 15
 - getClkSource, 15
 - getNumChannels, 15
 - getNumLines, 15
 - getSampleRate, 15
 - getSamplesPerChannel, 16
 - init, 16
 - initialized, 17
 - loops, 17
 - mode, 17
 - sampleRate, 18
 - samplesPerChannel, 18
 - setClkSource, 16
 - setSampleRate, 16
 - setSamplesPerChannel, 16
 - start, 16
 - status, 17
 - stop, 17
 - taskRef, 18
 - waitUntilDone, 17
 - writeToBuffer, 17
- divisor
 - Timing::Timing, 35
- do
 - WaveformGenerator::WaveformGenerator, 44
- eMapFilePath
 - itfParser::itfParser, 25
- eMapParser.eMapParser, 18
- eMapParser::eMapParser
 - read, 19
- eMapRead
 - itfParser::itfParser, 24
- eMapReadLine
 - itfParser::itfParser, 24
- eMapReadLines
 - itfParser::itfParser, 25
- edgeCntrTask
 - pCounter::pCounter, 32
- edgeCntrTerm
 - pCounter::pCounter, 32
- edgeCounter
 - pCounter::pCounter, 32
- empty
 - fileParser::fileParser, 21
 - itfParser::itfParser, 26
- enableStartTrigger
 - pCounter::pCounter, 32
- estAcqTime
 - AnalogOutput::AnalogOutput, 10
- fileObj
 - fileParser::fileParser, 21
- fileParser.fileParser, 19
- fileParser.fileParserError, 21
- fileParser::fileParser
 - __init__, 20
 - close, 20
 - comments, 21
 - empty, 21
 - fileObj, 21
 - getNumLines, 20
 - meta, 21
 - open, 21
 - totalLines, 21
- fileParser::fileParserError
 - __init__, 22
 - __str__, 22
 - msg, 22
- Finite
 - DAQmxUtility::Mode, 27
- gens
 - WaveformChassis::WaveformChassis, 40
- getClkSource
 - AnalogOutput::AnalogOutput, 7
 - DigitalOutput::DigitalOutput, 15
- getNumAoChannels
 - WaveformChassis::WaveformChassis, 38
- getNumChannels
 - AnalogOutput::AnalogOutput, 8
 - DigitalOutput::DigitalOutput, 15
- getNumDoChannels
 - WaveformChassis::WaveformChassis, 39
- getNumLines
 - DigitalOutput::DigitalOutput, 15
 - fileParser::fileParser, 20
- getSampleRate
 - AnalogOutput::AnalogOutput, 8
 - DigitalOutput::DigitalOutput, 15
- getSamplesPerChannel
 - AnalogOutput::AnalogOutput, 8
 - DigitalOutput::DigitalOutput, 16
- getStartTriggerSource

- AnalogOutput::AnalogOutput, 8
- Hardware
 - DAQmxUtility::TriggerType, 36
- init
 - AnalogOutput::AnalogOutput, 8
 - DigitalOutput::DigitalOutput, 16
 - pCounter::pCounter, 30
 - Timing::Timing, 34
 - WaveformChassis::WaveformChassis, 39
 - WaveformGenerator::WaveformGenerator, 43
- initFromFile
 - pCounter::pCounter, 30
 - WaveformChassis::WaveformChassis, 39
- initialized
 - AnalogOutput::AnalogOutput, 10
 - DigitalOutput::DigitalOutput, 17
 - Timing::Timing, 35
- itfParser.itfParser, 23
- itfParser::itfParser
 - __init__, 24
 - appendline, 24
 - comments, 25
 - eMapFilePath, 25
 - eMapRead, 24
 - eMapReadLine, 24
 - eMapReadLines, 25
 - empty, 26
 - meta, 26
 - read, 25
 - readline, 25
 - readlines, 25
 - tableHeader, 26
- loops
 - AnalogOutput::AnalogOutput, 10
 - DigitalOutput::DigitalOutput, 17
 - WaveformChassis::WaveformChassis, 41
 - WaveformGenerator::WaveformGenerator, 44
- measure
 - pCounter::pCounter, 31
- meta
 - fileParser::fileParser, 21
 - itfParser::itfParser, 26
- mode
 - AnalogOutput::AnalogOutput, 10
 - DigitalOutput::DigitalOutput, 17
 - WaveformChassis::WaveformChassis, 41
 - WaveformGenerator::WaveformGenerator, 44
- msg
 - fileParser::fileParserError, 22
 - niSyncError::niSyncError, 28
- niSyncError.niSyncError, 27
- niSyncError::niSyncError
 - __init__, 27
 - __str__, 28
- code, 28
- msg, 28
- numChannels
 - AnalogOutput::AnalogOutput, 11
- open
 - fileParser::fileParser, 21
- pCounter.pCounter, 28
- pCounter::pCounter
 - __del__, 30
 - __init__, 30
 - acqTime, 32
 - binTime, 32
 - clockCntrTask, 31
 - clockCounter, 31
 - clockSourceTerm, 31
 - close, 30
 - edgeCntrTask, 32
 - edgeCntrTerm, 32
 - edgeCounter, 32
 - enableStartTrigger, 32
 - init, 30
 - initFromFile, 30
 - measure, 31
 - read, 31
 - sampleRate, 32
 - samples, 33
 - start, 31
 - stop, 31
 - timeout, 32
 - triggerSource, 32
- pxiStarSlots
 - Timing::Timing, 35
- read
 - chassisConfigParser::chassisConfigParser, 12
 - eMapParser::eMapParser, 19
 - itfParser::itfParser, 25
 - pCounter::pCounter, 31
- readline
 - itfParser::itfParser, 25
- readlines
 - itfParser::itfParser, 25
- resourceName
 - Timing::Timing, 35
- sampleRate
 - AnalogOutput::AnalogOutput, 11
 - DigitalOutput::DigitalOutput, 18
 - pCounter::pCounter, 32
 - Timing::Timing, 35
 - WaveformChassis::WaveformChassis, 41
 - WaveformGenerator::WaveformGenerator, 45
- samples
 - pCounter::pCounter, 33
- samplesPerChannel
 - AnalogOutput::AnalogOutput, 11
 - DigitalOutput::DigitalOutput, 18

- WaveformChassis::WaveformChassis, 41
- WaveformGenerator::WaveformGenerator, 45
- sendSoftwareTrigger
 - Timing::Timing, 34
- session
 - Timing::Timing, 35
- setClkSource
 - AnalogOutput::AnalogOutput, 8
 - DigitalOutput::DigitalOutput, 16
- setSampleRate
 - AnalogOutput::AnalogOutput, 9
 - DigitalOutput::DigitalOutput, 16
- setSamplesPerChannel
 - AnalogOutput::AnalogOutput, 9
 - DigitalOutput::DigitalOutput, 16
- setStartTriggerSource
 - AnalogOutput::AnalogOutput, 9
- Software
 - DAQmxUtility::TriggerType, 36
- start
 - AnalogOutput::AnalogOutput, 9
 - DigitalOutput::DigitalOutput, 16
 - pCounter::pCounter, 31
 - WaveformChassis::WaveformChassis, 39
 - WaveformGenerator::WaveformGenerator, 43
- startTriggerSource
 - AnalogOutput::AnalogOutput, 11
 - WaveformChassis::WaveformChassis, 41
 - WaveformGenerator::WaveformGenerator, 45
- startTriggerSyncCard
 - AnalogOutput::AnalogOutput, 10
- Static
 - DAQmxUtility::Mode, 27
- status
 - AnalogOutput::AnalogOutput, 10
 - DigitalOutput::DigitalOutput, 17
 - Timing::Timing, 35
- stop
 - AnalogOutput::AnalogOutput, 9
 - DigitalOutput::DigitalOutput, 17
 - pCounter::pCounter, 31
 - WaveformChassis::WaveformChassis, 39
 - WaveformGenerator::WaveformGenerator, 43
- tableHeader
 - itfParser::itfParser, 26
- taskRef
 - AnalogOutput::AnalogOutput, 10
 - DigitalOutput::DigitalOutput, 18
- timeout
 - pCounter::pCounter, 32
- timing
 - WaveformChassis::WaveformChassis, 41
- Timing.Timing, 33
- Timing::Timing
 - __del__, 34
 - __init__, 34
 - close, 34
 - ddsFreq, 35
 - divisor, 35
 - init, 34
 - initialized, 35
 - pxiStarSlots, 35
 - resourceName, 35
 - sampleRate, 35
 - sendSoftwareTrigger, 34
 - session, 35
 - status, 35
- totalLines
 - fileParser::fileParser, 21
- triggerSource
 - pCounter::pCounter, 32
- triggerType
 - AnalogOutput::AnalogOutput, 11
 - WaveformChassis::WaveformChassis, 41
 - WaveformGenerator::WaveformGenerator, 45
- useAo
 - WaveformGenerator::WaveformGenerator, 45
- useDo
 - WaveformGenerator::WaveformGenerator, 45
- useTiming
 - WaveformChassis::WaveformChassis, 41
- waitUntilDone
 - AnalogOutput::AnalogOutput, 9
 - DigitalOutput::DigitalOutput, 17
 - WaveformChassis::WaveformChassis, 40
 - WaveformGenerator::WaveformGenerator, 44
- WaveformChassis.WaveformChassis, 36
- WaveformChassis::WaveformChassis
 - __init__, 38
 - clkSource, 40
 - close, 38
 - createAoSineBuffer, 38
 - createDoTestBuffer, 38
 - gens, 40
 - getNumAoChannels, 38
 - getNumDoChannels, 39
 - init, 39
 - initFromFile, 39
 - loops, 41
 - mode, 41
 - sampleRate, 41
 - samplesPerChannel, 41
 - start, 39
 - startTriggerSource, 41
 - stop, 39
 - timing, 41
 - triggerType, 41
 - useTiming, 41
 - waitUntilDone, 40
 - writeAoBuffer, 40
 - writeDoBuffer, 40
 - writeStartStop, 40
- WaveformGenerator.WaveformGenerator, 42
- WaveformGenerator.writeStartStop, 45
- WaveformGenerator::WaveformGenerator

- [__init__, 43](#)
 - [ao, 44](#)
 - [clkSource, 44](#)
 - [close, 43](#)
 - [do, 44](#)
 - [init, 43](#)
 - [loops, 44](#)
 - [mode, 44](#)
 - [sampleRate, 45](#)
 - [samplesPerChannel, 45](#)
 - [start, 43](#)
 - [startTriggerSource, 45](#)
 - [stop, 43](#)
 - [triggerType, 45](#)
 - [useAo, 45](#)
 - [useDo, 45](#)
 - [waitUntilDone, 44](#)
 - [writeAoBuffer, 44](#)
 - [writeDoBuffer, 44](#)
- write
 - [chassisConfigParser::chassisConfigParser, 12](#)
- writeAoBuffer
 - [WaveformChassis::WaveformChassis, 40](#)
 - [WaveformGenerator::WaveformGenerator, 44](#)
- writeDoBuffer
 - [WaveformChassis::WaveformChassis, 40](#)
 - [WaveformGenerator::WaveformGenerator, 44](#)
- writeStartStop
 - [WaveformChassis::WaveformChassis, 40](#)
- writeToBuffer
 - [AnalogOutput::AnalogOutput, 10](#)
 - [DigitalOutput::DigitalOutput, 17](#)