

---

# **MSL-Equipment Documentation**

*Release 0.1.0*

**Measurement Standards Laboratory**

**Jul 13, 2017**



---

## Contents

---

<b>1</b>	<b>Install MSL-Equipment</b>	<b>3</b>
<b>2</b>	<b>Example Usage</b>	<b>5</b>
<b>3</b>	<b>MSL-Equipment API Documentation</b>	<b>7</b>
3.1	Package Structure . . . . .	7
<b>4</b>	<b>MSL-Equipment Database Formats</b>	<b>197</b>
4.1	Equipment-Register Database . . . . .	197
4.2	Connections Database . . . . .	199
<b>5</b>	<b>MSL Resources</b>	<b>201</b>
5.1	SDK Wrapper Classes . . . . .	201
<b>6</b>	<b>License</b>	<b>203</b>
<b>7</b>	<b>Developers</b>	<b>205</b>
<b>8</b>	<b>Release Notes</b>	<b>207</b>
8.1	Version 0.1.0 . . . . .	207
<b>9</b>	<b>Index</b>	<b>209</b>
	<b>Python Module Index</b>	<b>211</b>



Manage and connect to equipment in the laboratory.



# CHAPTER 1

---

## Install MSL-Equipment

---

To install **MSL-Equipment** run:

```
pip install https://github.com/MSLNZ/msl-equipment/archive/master.zip
```





## CHAPTER 2

---

### Example Usage

---

The following illustrates how to use **MSL-Equipment**.



---

## MSL-Equipment API Documentation

---

The root package is

<i>mssl.equipment</i>	Manage and connect to equipment in the laboratory.
-----------------------	--

which has the following user-facing modules to manage and connect to equipment

<i>mssl.equipment.config</i>	Load a XML configuration file.
<i>mssl.equipment.database</i>	Load equipment and connection records (rows) from databases.
<i>mssl.equipment.factory</i>	Establish a connection to the equipment to send and receive messages.

and the following module that contains the package constants

<i>mssl.equipment.constants</i>	MSL-Equipment package constants.
---------------------------------	----------------------------------

## Package Structure

### **mssl.equipment package**

Manage and connect to equipment in the laboratory.

`mssl.equipment.version_info = version_info(major=0, minor=1, micro=0)`  
*namedtuple* – Contains the version information as a (major, minor, micro) tuple.

### **Subpackages**

### msl.equipment.resources package

MSL resources for connecting to equipment.

`msl.equipment.resources.recursive_find_resource_class` (*class\_name*)

Find the Python resource class.

**Parameters** `class_name` (*str*) – The name of a resource class.

**Returns** The Python resource class (a subclass of `Connection`).

**Raises** `IOError` – If the Python resource class cannot be found.

`msl.equipment.resources.get_class` (*module\_name*, *class\_name*)

Returns the specified Python class.

**Parameters**

- `module_name` (*str*) – The name of a Python module.
- `class_name` (*str*) – The name of a Python class.

**Returns** The Python resource class or `None` if the class cannot be found.

`msl.equipment.resources.check_manufacture_model_resource_name` (*connection\_record*)

Check if there is a resource class with the name equal to `connection_record.model` in the `connection_record.manufacturer.lower() + '.' + connection_record.model.lower()` module.

For example, if the `connection_record` is for a Thorlabs FW102C filter wheel then check if a `msl.equipment.resources.thorlabs.fw102C.FW102C` resource class exists.

**Parameters** `connection_record` (*msl.equipment.record\_types.ConnectionRecord*) – A *msl.equipment.record\_types.ConnectionRecord* object.

**Returns** The Python resource class or `None` if there is no resource class available.

`msl.equipment.resources.find_sdk_class` (*connection\_record*)

Find the Python class that is a wrapper around the SDK.

**Parameters** `connection_record` (*msl.equipment.record\_types.ConnectionRecord*) – A *msl.equipment.record\_types.ConnectionRecord* object.

**Returns** *The Python wrapper class around the manufacturer's SDK.*

**Raises** `IOError` – If the `msl.equipment.record_types.ConnectionRecord.address` value does not have the required format of `SDK::PythonClassName::PathToLibrary`.

`msl.equipment.resources.find_serial_class` (*connection\_record*)

Find the Python resource class that is used for `Serial` communication.

**Parameters** `connection_record` (*msl.equipment.record\_types.ConnectionRecord*) – A *msl.equipment.record\_types.ConnectionRecord* object.

**Returns**

- The Python resource class that uses `Serial` communication or
- `None` if no resource class was specified in the address.

## Subpackages

**msl.equipment.resources.bentham** package

## Submodules

**msl.equipment.resources.bentham.benhw32** module

A wrapper around the Bentham SDK `benhw32_cdecl.dll`.

**class** `msl.equipment.resources.bentham.benhw32.Bentham32` (*host, port, quiet, \*\*kwargs*)

Bases: `msl.loadlib.server32.Server32`

A wrapper around the Bentham SDK `benhw32_cdecl.dll`.

Do not instantiate this class directly. Use `benhw64.Bentham`.

**auto\_measure** ()

**auto\_range** ()

**build\_group** ()

**build\_system\_model** (*path*)

**close** ()

**close\_shutter** ()

**component\_select\_wl** (*p\_id, wavelength*)

**get** (*hw\_id, token, index*)

**get\_c\_group** (*n*)

**get\_component\_list** ()

**get\_group** (*group*)

**get\_hardware\_type** (*hw\_id*)

**get\_mono\_items** (*hw\_id*)

**get\_no\_of\_dark\_currents** ()

**get\_zero\_calibration\_info** ()

**group\_add** (*p\_id, group*)

**group\_remove** (*p\_id, group*)

**initialise** ()

**load\_setup** (*path*)

**measurement** ()

**multi\_auto\_range** ()

**multi\_get\_no\_of\_dark\_currents** (*group*)

**multi\_get\_zero\_calibration\_info** (*group*)

**multi\_initialise** ()  
**multi\_measurement** ()  
**multi\_select\_wavelength** (*wavelength*)  
**multi\_zero\_calibration** (*start\_wavelength, stop\_wavelength*)  
**park** ()  
**read** (*p\_message, buffer\_size, p\_id*)  
**report\_error** ()  
**save\_setup** (*p\_file\_name*)  
**select\_wavelength** (*wavelength*)  
**send** (*p\_message, p\_id*)  
**set** (*hw\_id, token, index, value*)  
**trace** (*on*)  
**use\_group** (*group*)  
**get\_version** ()  
**zero\_calibration** (*start\_wavelength, stop\_wavelength*)  
**camera\_get\_zero\_calibration\_info** (*p\_id*)  
**camera\_measurement** (*p\_id, num*)  
**camera\_zero\_calibration** (*p\_id, start\_wavelength, stop\_wavelength*)  
**delete\_group** (*n*)  
**display\_advanced\_window** (*p\_id, hinstance*)  
**display\_setup\_window** (*p\_id, hinstance*)  
**get\_log** (*log*)  
**get\_log\_size** ()  
**get\_max\_bw** (*group, start\_wavelength, stop\_wavelength*)  
**get\_min\_step** (*group, start\_wavelength, stop\_wavelength*)  
**get\_n\_groups** ()  
**get\_str** (*hw\_id, token, index, s*)  
**mapped\_logging** (*i*)  
**multi\_auto\_measure** ()  
**multi\_park** ()  
**start\_log** (*c\_list*)  
**stop\_log** (*c\_list*)

**msl.equipment.resources.bentham.benhw64 module**

A wrapper around the Bentham SDK `benhw32_cdecl.dll`.

**class** `msl.equipment.resources.bentham.benhw64.Bentham` (*record*)

Bases: `msl.equipment.connection.Connection`, `msl.loadlib.client64.Client64`

A wrapper around the `benhw32.Bentham32` class.

This class can be used with either a 32-bit or 64-bit Python interpreter to call the 32-bit functions in `benhw32_cdecl.dll`.

The `record.connection.properties` dictionary for a Bentham device supports the following key-value pairs:

```
'model': 'C:\path\to\System.cfg', # default is ''
'setup': 'C:\path\to\System.atr', # default is ''
```

If both properties are not defined in the **Connections Database** then you will have to call `build_system_model()`, `load_setup()` and `initialise()` to configure the SDK.

Do not instantiate this class directly. Use the factory method, `msl.equipment.factory.connect`, or the *record* object itself, `record.connect()`, to connect to the equipment.

**Parameters** *record* (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**auto\_measure** ()

**build\_system\_model** (*path*)

**disconnect** ()

**errcheck** (*result*, \**args*, *append\_msg*='')

**get** (*hw\_id*, *token*, *index*)

**get\_component\_list** ()

**get\_hardware\_type** (*hw\_id*)

**get\_mono\_items** (*hw\_id*)

**wavelength**

**initialise** ()

**load\_setup** (*path*)

**park** ()

**select\_wavelength** (*wavelength*)

**set** (*hw\_id*, *token*, *index*, *value*)

**version** ()

**zero\_calibration** (*start\_wavelength*, *stop\_wavelength*)

### **msl.equipment.resources.bentham.errors module**

Error code definition file for Bentham Instruments Spectroradiometer Control DLL

### **msl.equipment.resources.bentham.tokens module**

Attribute token definition file for Bentham Instruments Spectroradiometer Control DLL.

### **msl.equipment.resources.picotech package**

Resources for equipment from [Pico Technology](#).

### **Subpackages**

#### **msl.equipment.resources.picotech.picoscope package**

Wrapper around the PicoScope SDK from Pico Technologies.

This package was initially created using Pico Technology SDK 64-bit v10.6.10.24

The latest SDK can be downloaded from [here](#).

### **Submodules**

#### **msl.equipment.resources.picotech.picoscope.callbacks module**

Callback functions in the Pico Technology SDK v10.6.10.24

#### **msl.equipment.resources.picotech.picoscope.channel module**

Contains the information about a PicoScope channel.

```
class msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel (channel,  
en-  
abled,  
cou-  
pling,  
volt-  
age_range,  
volt-  
age_offset,  
band-  
width,  
max_adu_va
```

Bases: `object`

Contains the information about a PicoScope channel.

This class is used by `picoscope.PicoScope` and is not meant to be called directly.



**Parameters**

- **channel** (`enum.IntEnum`) – The channel number.
- **enabled** (`bool`) – Whether the channel is enabled.
- **coupling** (`enum.IntEnum`) – The coupling, e.g. AC or DC.
- **voltage\_range** (`float`) – The voltage range, in Volts.
- **voltage\_offset** (`float`) – The voltage offset, in Volts.
- **bandwidth** (`enum.IntEnum` or `None`) – The bandwidth used, if the PicoScope supports a `BandwidthLimiter`.
- **max\_adu\_value** (`int`) – The maximum analog-to-digital unit.

**channel**

`enum.IntEnum` – The channel number.

**enabled**

`bool` – Whether the channel is enabled.

**coupling**

`enum.IntEnum` – The coupling, e.g. AC or DC.

**voltage\_range**

`float` – The voltage range, in Volts.

**voltage\_offset**

`float` – The voltage offset, in Volts.

**bandwidth**

`enum.IntEnum` or `None` – The bandwidth used, if the PicoScope supports a `BandwidthLimiter`.

**volts\_per\_adu**

`float` – The conversion factor to convert ADU to volts

**raw**

`numpy.ndarray` – The raw data, in ADU

**buffer**

`numpy.ndarray` – The raw data, in ADU

**volts**

`numpy.ndarray` – The data, in volts

**num\_samples**

`int` – The size of the data array.

**allocate** (`num_captures`, `num_samples`)

Allocate memory to save the data.

**Parameters**

- **num\_captures** (`int`) – The number of captures
- **num\_samples** (`int`) – The number of samples

## **msl.equipment.resources.picotech.picoscope.enums module**

Enums defined in the Pico Technology SDK v10.6.10.24

**class** `msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfoApi`

Bases: `enum.IntEnum`

Constants that can be passed to the `get_unit_info()` method.

**DRIVER\_VERSION = 0**

**USB\_VERSION = 1**

**HARDWARE\_VERSION = 2**

**VARIANT\_INFO = 3**

**BATCH\_AND\_SERIAL = 4**

**CAL\_DATE = 5**

**KERNEL\_VERSION = 6**

**DIGITAL\_HARDWARE\_VERSION = 7**

**ANALOGUE\_HARDWARE\_VERSION = 8**

**FIRMWARE\_VERSION\_1 = 9**

**FIRMWARE\_VERSION\_2 = 10**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000Channel`

Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**NONE = 5**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000Range`

Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**

**R\_20MV = 1**

**R\_50MV = 2**

**R\_100MV = 3**

**R\_200MV = 4**

**R\_500MV = 5**

**R\_1V = 6**  
**R\_2V = 7**  
**R\_5V = 8**  
**R\_10V = 9**  
**R\_20V = 10**  
**R\_50V = 11**  
**R\_MAX = 12**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000TimeUnits`

Bases: `enum.IntEnum`

An enumeration.

**FS = 0**  
**PS = 1**  
**NS = 2**  
**US = 3**  
**MS = 4**  
**S = 5**  
**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000Error`

Bases: `enum.IntEnum`

An enumeration.

**OK = 0**  
**MAX\_UNITS\_OPENED = 1**  
**MEM\_FAIL = 2**  
**NOT\_FOUND = 3**  
**FW\_FAIL = 4**  
**NOT\_RESPONDING = 5**  
**CONFIG\_FAIL = 6**  
**OS\_NOT\_SUPPORTED = 7**  
**PICOPP\_TOO\_OLD = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000Info`

Bases: `enum.IntEnum`

An enumeration.

**DRIVER\_VERSION = 0**  
**USB\_VERSION = 1**  
**HARDWARE\_VERSION = 2**

**VARIANT\_INFO = 3**

**BATCH\_AND\_SERIAL = 4**

**CAL\_DATE = 5**

**ERROR\_CODE = 6**

**KERNEL\_DRIVER\_VERSION = 7**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000TriggerDirection`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**MAX = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000OpenProgress`

Bases: `enum.IntEnum`

An enumeration.

**FAIL = -1**

**PENDING = 0**

**COMPLETE = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000EtsMode`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ButtonState`

Bases: `enum.IntEnum`

An enumeration.

**NO\_PRESS = 0**

**SHORT\_PRESS = 1**

**LONG\_PRESS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000SweepType`

Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000WaveType`

Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**

**RAMPUP = 3**

**RAMPDOWN = 4**

**DC\_VOLTAGE = 5**

**GAUSSIAN = 6**

**SINC = 7**

**HALF\_SINE = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ThresholdDirection`

Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**ADV\_RISING = 2**

**ADV\_FALLING = 3**

**RISING\_OR\_FALLING = 4**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**ADV\_NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000TriggerState`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000PulseWidthType`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex`

Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**MAX = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel`

Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**AUX = 5**

**MAX\_TRIGGER\_SOURCES = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ATriggerOperand`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**OR = 1**

**AND = 2**

**THEN = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000DigitalPort`

Bases: `enum.IntEnum`

An enumeration.

**PORT0 = 128**

**PORT1 = 129**

**PORT2 = 130**

**PORT3 = 131**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel`

Bases: `enum.IntEnum`

An enumeration.

**CHANNEL\_0 = 0**

**CHANNEL\_1 = 1**

**CHANNEL\_2 = 2**

**CHANNEL\_3 = 3**

**CHANNEL\_4 = 4**

**CHANNEL\_5 = 5**

**CHANNEL\_6 = 6**

**CHANNEL\_7 = 7**

**CHANNEL\_8 = 8**

**CHANNEL\_9 = 9**

**CHANNEL\_10 = 10**

**CHANNEL\_11 = 11**

**CHANNEL\_12 = 12**

**CHANNEL\_13 = 13**

**CHANNEL\_14 = 14**

**CHANNEL\_15 = 15**

**CHANNEL\_16 = 16**

```
CHANNEL_17 = 17
CHANNEL_18 = 18
CHANNEL_19 = 19
CHANNEL_20 = 20
CHANNEL_21 = 21
CHANNEL_22 = 22
CHANNEL_23 = 23
CHANNEL_24 = 24
CHANNEL_25 = 25
CHANNEL_26 = 26
CHANNEL_27 = 27
CHANNEL_28 = 28
CHANNEL_29 = 29
CHANNEL_30 = 30
CHANNEL_31 = 31
CHANNEL_MAX = 32
```

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ARange`

Bases: `enum.IntEnum`

An enumeration.

```
R_10MV = 0
R_20MV = 1
R_50MV = 2
R_100MV = 3
R_200MV = 4
R_500MV = 5
R_1V = 6
R_2V = 7
R_5V = 8
R_10V = 9
R_20V = 10
R_50V = 11
R_MAX = 12
```

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ACoupling`

Bases: `enum.IntEnum`

An enumeration.



**AC = 0**

**DC = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelInfo`

Bases: `enum.IntEnum`

An enumeration.

**RANGES = 0**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AETSMode`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ATimeUnits`

Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ASweepType`

Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType`

Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**  
**RAMP\_UP = 3**  
**RAMP\_DOWN = 4**  
**SINC = 5**  
**GAUSSIAN = 6**  
**HALF\_SINE = 7**  
**DC\_VOLTAGE = 8**  
**MAX = 9**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AExtraOperations`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**  
**WHITENOISE = 1**  
**PRBS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ASigGenTrigType`  
Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**  
**FALLING = 1**  
**GATE\_HIGH = 2**  
**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ASigGenTrigSource`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**  
**SCOPE\_TRIG = 1**  
**AUX\_IN = 2**  
**EXT\_IN = 3**  
**SOFT\_TRIG = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AIndexMode`  
Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**  
**DUAL = 1**  
**QUAD = 2**  
**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000A_ThresholdMode`  
Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AThresholdDirection`  
Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**ABOVE\_LOWER = 5**

**BELOW\_LOWER = 6**

**RISING\_LOWER = 7**

**FALLING\_LOWER = 8**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**POSITIVE\_RUNT = 9**

**NEGATIVE\_RUNT = 10**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection`  
Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**LOW = 1**

**HIGH = 2**

**RISING = 3**

**FALLING = 4**

**RISING\_OR\_FALLING = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ATriggerState`  
Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000ARatioMode`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**DECIMATE = 2**

**AVERAGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000APulseWidthType`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldOffType`  
Bases: `enum.IntEnum`

An enumeration.

**TIME = 0**

**MAX = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000Channel`  
Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**NONE = 5**

**MAX\_TRIGGER\_SOURCES = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000Range`

Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**

**R\_20MV = 1**

**R\_50MV = 2**

**R\_100MV = 3**

**R\_200MV = 4**

**R\_500MV = 5**

**R\_1V = 6**

**R\_2V = 7**

**R\_5V = 8**

**R\_10V = 9**

**R\_20V = 10**

**R\_50V = 11**

**R\_100V = 12**

**R\_200V = 13**

**R\_400V = 14**

**R\_MAX = 15**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000WaveTypes`

Bases: `enum.IntEnum`

An enumeration.

**SQUARE = 0**

**TRIANGLE = 1**

**SINE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000TimeUnits`

Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000Error`

Bases: `enum.IntEnum`

An enumeration.

**OK = 0**

**MAX\_UNITS\_OPENED = 1**

**MEM\_FAIL = 2**

**NOT\_FOUND = 3**

**FW\_FAIL = 4**

**NOT\_RESPONDING = 5**

**CONFIG\_FAIL = 6**

**OS\_NOT\_SUPPORTED = 7**

**PICOPP\_TOO\_OLD = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000Info`

Bases: `enum.IntEnum`

An enumeration.

**DRIVER\_VERSION = 0**

**USB\_VERSION = 1**

**HARDWARE\_VERSION = 2**

**VARIANT\_INFO = 3**

**BATCH\_AND\_SERIAL = 4**

**CAL\_DATE = 5**

**ERROR\_CODE = 6**

**KERNEL\_DRIVER\_VERSION = 7**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerDirection`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**MAX = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000OpenProgress`

Bases: `enum.IntEnum`

An enumeration.

**FAIL = -1**

**PENDING = 0**

**COMPLETE = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000EtsMode`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdDirection`

Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerState`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000PulseWidthType`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ABandwidthLimiter`  
Bases: `enum.IntEnum`

An enumeration.

**BW\_FULL = 0**

**BW\_20MHZ = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex`  
Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**MAX = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel`  
Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**AUX = 5**

**MAX\_TRIGGER\_SOURCES = 6**



**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort`

Bases: `enum.IntEnum`

An enumeration.

**PORT0 = 128**

**PORT1 = 129**

**PORT2 = 130**

**PORT3 = 131**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel`

Bases: `enum.IntEnum`

An enumeration.

**CHANNEL\_0 = 0**

**CHANNEL\_1 = 1**

**CHANNEL\_2 = 2**

**CHANNEL\_3 = 3**

**CHANNEL\_4 = 4**

**CHANNEL\_5 = 5**

**CHANNEL\_6 = 6**

**CHANNEL\_7 = 7**

**CHANNEL\_8 = 8**

**CHANNEL\_9 = 9**

**CHANNEL\_10 = 10**

**CHANNEL\_11 = 11**

**CHANNEL\_12 = 12**

**CHANNEL\_13 = 13**

**CHANNEL\_14 = 14**

**CHANNEL\_15 = 15**

**CHANNEL\_16 = 16**

**CHANNEL\_17 = 17**

**CHANNEL\_18 = 18**

**CHANNEL\_19 = 19**

**CHANNEL\_20 = 20**

**CHANNEL\_21 = 21**

**CHANNEL\_22 = 22**

**CHANNEL\_23 = 23**

```
CHANNEL_24 = 24
CHANNEL_25 = 25
CHANNEL_26 = 26
CHANNEL_27 = 27
CHANNEL_28 = 28
CHANNEL_29 = 29
CHANNEL_30 = 30
CHANNEL_31 = 31
CHANNEL_MAX = 32
```

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ARange
    Bases: enum.IntEnum
```

An enumeration.

```
R_10MV = 0
R_20MV = 1
R_50MV = 2
R_100MV = 3
R_200MV = 4
R_500MV = 5
R_1V = 6
R_2V = 7
R_5V = 8
R_10V = 9
R_20V = 10
R_50V = 11
R_MAX = 12
```

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ACoupling
    Bases: enum.IntEnum
```

An enumeration.

```
AC = 0
DC = 1
```

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelInfo
    Bases: enum.IntEnum
```

An enumeration.

```
RANGES = 0
```

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AetsMode`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ATimeUnits`

Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ASweepType`

Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType`

Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**

**RAMP\_UP = 3**

**RAMP\_DOWN = 4**

**SINC = 5**

**GAUSSIAN = 6**

**HALF\_SINE = 7**

**DC\_VOLTAGE = 8**

**MAX = 9**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AExtraOperations`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**WHITENOISE = 1**

**PRBS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ASigGenTrigType`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH = 2**

**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ASigGenTrigSource`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**SCOPE\_TRIG = 1**

**AUX\_IN = 2**

**EXT\_IN = 3**

**SOFT\_TRIG = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AIndexMode`

Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**

**DUAL = 1**

**QUAD = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000A_ThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection`

Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**ABOVE\_LOWER = 5**

**BELOW\_LOWER = 6**

**RISING\_LOWER = 7**

**FALLING\_LOWER = 8**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**POSITIVE\_RUNT = 9**

**NEGATIVE\_RUNT = 10**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalDirection`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**LOW = 1**

**HIGH = 2**

**RISING = 3**

**FALLING = 4**

**RISING\_OR\_FALLING = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ATriggerState`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000ARatioMode`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**DECIMATE = 2**

**AVERAGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000APulseWidthType`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldOffType`

Bases: `enum.IntEnum`

An enumeration.

**TIME = 0**

**EVENT = 1**

**MAX = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex`

Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**MAX = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000Channel`

Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**AUX = 5**

**MAX\_TRIGGER\_SOURCES = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000Range`

Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**

**R\_20MV = 1**

**R\_50MV = 2**

**R\_100MV = 3**

**R\_200MV = 4**

**R\_500MV = 5**

**R\_1V = 6**

**R\_2V = 7**

**R\_5V = 8**

**R\_10V = 9**

**R\_20V = 10**

**R\_50V = 11**

**R\_100V = 12**

**MAX\_RANGES = 13**

**RESISTANCE\_100R = 13**

**RESISTANCE\_1K = 14**

**RESISTANCE\_10K = 15**

**RESISTANCE\_100K = 16**

**RESISTANCE\_1M = 17**

**MAX\_RESISTANCES = 18**

**ACCELEROMETER\_10MV = 18**

ACCELEROMETER\_20MV = 19  
ACCELEROMETER\_50MV = 20  
ACCELEROMETER\_100MV = 21  
ACCELEROMETER\_200MV = 22  
ACCELEROMETER\_500MV = 23  
ACCELEROMETER\_1V = 24  
ACCELEROMETER\_2V = 25  
ACCELEROMETER\_5V = 26  
ACCELEROMETER\_10V = 27  
ACCELEROMETER\_20V = 28  
ACCELEROMETER\_50V = 29  
ACCELEROMETER\_100V = 30  
MAX\_ACCELEROMETER = 31  
TEMPERATURE\_UPTO\_40 = 31  
TEMPERATURE\_UPTO\_70 = 32  
TEMPERATURE\_UPTO\_100 = 33  
TEMPERATURE\_UPTO\_130 = 34  
MAX\_TEMPERATURES = 35  
RESISTANCE\_5K = 35  
RESISTANCE\_25K = 36  
RESISTANCE\_50K = 37  
MAX\_EXTRA\_RESISTANCES = 38

**class** msl.equipment.resources.picotech.picoscope.enums.**PS4000Probe**

Bases: `enum.IntEnum`

An enumeration.

NONE = 0  
CURRENT\_CLAMP\_10A = 1  
CURRENT\_CLAMP\_1000A = 2  
TEMPERATURE\_SENSOR = 3  
CURRENT\_MEASURING\_DEVICE = 4  
PRESSURE\_SENSOR\_50BAR = 5  
PRESSURE\_SENSOR\_5BAR = 6  
OPTICAL\_SWITCH = 7  
UNKNOWN = 8  
MAX\_PROBES = 8



**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelInfo`  
Bases: `enum.IntEnum`

An enumeration.

**RANGES = 0**

**RESISTANCES = 1**

**ACCELEROMETER = 2**

**PROBES = 3**

**TEMPERATURES = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000EtsMode`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000TimeUnits`  
Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000SweepType`  
Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000OperationTypes`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**WHITENOISE = 1**

**PRBS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType`

Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**

**RAMP\_UP = 3**

**RAMP\_DOWN = 4**

**SINC = 5**

**GAUSSIAN = 6**

**HALF\_SINE = 7**

**DC\_VOLTAGE = 8**

**WHITE\_NOISE = 9**

**MAX = 10**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000SigGenTrigType`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH = 2**

**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000SigGenTrigSource`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**SCOPE\_TRIG = 1**

**AUX\_IN = 2**

**EXT\_IN = 3**

**SOFT\_TRIG = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000IndexMode`

Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**

**DUAL = 1**

**QUAD = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection`

Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**ABOVE\_LOWER = 5**

**BELOW\_LOWER = 6**

**RISING\_LOWER = 7**

**FALLING\_LOWER = 8**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**POSITIVE\_RUNT = 9**

**NEGATIVE\_RUNT = 10**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000TriggerState`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000RatioMode`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**AVERAGE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000PulseWidthType`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000Ps4000HoldOffType`  
Bases: `enum.IntEnum`

An enumeration.

**TIME = 0**

**MAX = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000FrequencyCounterRange`  
Bases: `enum.IntEnum`

An enumeration.

**FC\_2K = 0**

**FC\_20K = 1**

**FC\_20 = 2**

**FC\_200 = 3**

**FC\_MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**WHITENOISE = 1**

**PRBS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ABandwidthLimiter`  
Bases: `enum.IntEnum`

An enumeration.

**BW\_FULL = 0**

**BW\_20KHZ = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ACoupling`

Bases: `enum.IntEnum`

An enumeration.

**AC = 0**

**DC = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel`

Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**MAX\_4\_CHANNELS = 4**

**E = 4**

**F = 5**

**G = 6**

**H = 7**

**EXT = 8**

**MAX\_CHANNELS = 8**

**AUX = 9**

**MAX\_TRIGGER\_SOURCES = 10**

**PULSE\_WIDTH\_SOURCE = 268435456**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex`

Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**E\_MAX = 8**  
**E\_MIN = 9**  
**F\_MAX = 10**  
**F\_MIN = 11**  
**G\_MAX = 12**  
**G\_MIN = 13**  
**H\_MAX = 14**  
**H\_MIN = 15**  
**MAX = 16**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ARange`  
Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**  
**R\_20MV = 1**  
**R\_50MV = 2**  
**R\_100MV = 3**  
**R\_200MV = 4**  
**R\_500MV = 5**  
**R\_1V = 6**  
**R\_2V = 7**  
**R\_5V = 8**  
**R\_10V = 9**  
**R\_20V = 10**  
**R\_50V = 11**  
**R\_100V = 12**  
**R\_200V = 13**  
**R\_MAX = 14**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AResistanceRange`  
Bases: `enum.IntEnum`

An enumeration.

**R\_315K = 512**  
**R\_1100K = 513**  
**R\_10M = 514**  
**R\_MAX = 3**  
**R\_ADCV = 268435456**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AetsMode`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ATimeUnits`

Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ASweepType`

Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType`

Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**

**RAMP\_UP = 3**

**RAMP\_DOWN = 4**

**SINC = 5**

**GAUSSIAN = 6**

**HALF\_SINE = 7**

**DC\_VOLTAGE = 8**

**WHITE\_NOISE = 9**

**MAX = 10**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelLed`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**RED = 1**

**GREEN = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000MetaType`

Bases: `enum.IntEnum`

An enumeration.

**UNIT\_INFO = 0**

**DEVICE\_CAPABILITY = 1**

**DEVICE\_SETTINGS = 2**

**SIGNAL\_GENERATOR\_SETTINGS = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000MetaOperation`

Bases: `enum.IntEnum`

An enumeration.

**READ = 0**

**WRITE = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000MetaFormat`

Bases: `enum.IntEnum`

An enumeration.

**COMMA\_SEPERATED = 0**

**XML = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ASigGenTrigType`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH = 2**

**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ASigGenTrigSource`

Bases: `enum.IntEnum`

An enumeration.



**NONE = 0**

**SCOPE\_TRIG = 1**

**AUX\_IN = 2**

**EXT\_IN = 3**

**SOFT\_TRIG = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AIndexMode`

Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**

**DUAL = 1**

**QUAD = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection`

Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**ABOVE\_LOWER = 5**

**BELOW\_LOWER = 6**

**RISING\_LOWER = 7**

**FALLING\_LOWER = 8**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**POSITIVE\_RUNT = 9**

**NEGATIVE\_RUNT = 10**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerState`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ASensorState`

Bases: `enum.IntEnum`

An enumeration.

**CONNECT\_STATE\_FLOATING = 0**

**SENSOR\_STATE\_CONNECTED = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AFrequencyCounterRange`

Bases: `enum.IntEnum`

An enumeration.

**FC\_2K = 0**

**FC\_20K = 1**

**FC\_20 = 2**

**FC\_200 = 3**

**FC\_MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AConditionsInfo`

Bases: `enum.IntEnum`

An enumeration.

**CLEAR = 1**

**ADD = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**DECIMATE = 2**

**AVERAGE = 4**

**DISTRIBUTION = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000APulseWidthType`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelInfo`

Bases: `enum.IntEnum`

An enumeration.

**RANGES = 0**

**RESISTANCES = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue`

Bases: `enum.IntEnum`

An enumeration.

**MEMORY = 0**

**MEMORY\_NO\_OF\_SEGMENTS = 1**

**MEMORY\_MAX\_SAMPLES = 2**

**NO\_OF\_CHANNELS = 3**

**ARRAY\_OF\_CHANNELS = 4**

**CHANNEL = 5**

**CHANNEL\_NAME = 6**

**CHANNEL\_RANGE = 7**

**CHANNEL\_COUPLING = 8**

**CHANNEL\_ENABLED = 9**

**CHANNEL\_ANALOGUE\_OFFSET = 10**

**CHANNEL\_BANDWIDTH = 11**

**TRIGGER = 12**

**TRIGGER\_AUXIO\_OUTPUT\_ENABLED = 13**

**TRIGGER\_AUTO\_TRIGGER\_MILLISECONDS = 14**

**TRIGGER\_PROPERTIES = 15**

**NO\_OF\_TRIGGER\_PROPERTIES = 16**

**TRIGGER\_PROPERTIES\_CHANNEL = 17**

**TRIGGER\_PROPERTIES\_THRESHOLD\_UPPER = 18**

TRIGGER\_PROPERTIES\_THRESHOLD\_UPPER\_HYSTERESIS = 19  
TRIGGER\_PROPERTIES\_THRESHOLD\_LOWER = 20  
TRIGGER\_PROPERTIES\_THRESHOLD\_LOWER\_HYSTERESIS = 21  
TRIGGER\_PROPERTIES\_THRESHOLD\_MODE = 22  
TRIGGER\_ARRAY\_OF\_BLOCK\_CONDITIONS = 23  
TRIGGER\_NO\_OF\_BLOCK\_CONDITIONS = 24  
TRIGGER\_CONDITIONS = 25  
TRIGGER\_NO\_OF\_CONDITIONS = 26  
TRIGGER\_CONDITION\_SOURCE = 27  
TRIGGER\_CONDITION\_STATE = 28  
TRIGGER\_DIRECTION = 29  
TRIGGER\_NO\_OF\_DIRECTIONS = 30  
TRIGGER\_DIRECTION\_CHANNEL = 31  
TRIGGER\_DIRECTION\_DIRECTION = 32  
TRIGGER\_DELAY = 33  
TRIGGER\_DELAY\_MS = 34  
FREQUENCY\_COUNTER = 35  
FREQUENCY\_COUNTER\_ENABLED = 36  
FREQUENCY\_COUNTER\_CHANNEL = 37  
FREQUENCY\_COUNTER\_RANGE = 38  
FREQUENCY\_COUNTER\_TRESHOLDMAJOR = 39  
FREQUENCY\_COUNTER\_TRESHOLDMINOR = 40  
PULSE\_WIDTH\_PROPERTIES = 41  
PULSE\_WIDTH\_PROPERTIES\_DIRECTION = 42  
PULSE\_WIDTH\_PROPERTIES\_LOWER = 43  
PULSE\_WIDTH\_PROPERTIES\_UPPER = 44  
PULSE\_WIDTH\_PROPERTIES\_TYPE = 45  
PULSE\_WIDTH\_ARRAY\_OF\_BLOCK\_CONDITIONS = 46  
PULSE\_WIDTH\_NO\_OF\_BLOCK\_CONDITIONS = 47  
PULSE\_WIDTH\_CONDITIONS = 48  
PULSE\_WIDTH\_NO\_OF\_CONDITIONS = 49  
PULSE\_WIDTH\_CONDITIONS\_SOURCE = 50  
PULSE\_WIDTH\_CONDITIONS\_STATE = 51  
SAMPLE\_PROPERTIES = 52

SAMPLE\_PROPERTIES\_PRE\_TRIGGER\_SAMPLES = 53  
SAMPLE\_PROPERTIES\_POST\_TRIGGER\_SAMPLES = 54  
SAMPLE\_PROPERTIES\_TIMEBASE = 55  
SAMPLE\_PROPERTIES\_NO\_OF\_CAPTURES = 56  
SAMPLE\_PROPERTIES\_RESOLUTION = 57  
SAMPLE\_PROPERTIES\_OVERLAPPED = 58  
SAMPLE\_PROPERTIES\_OVERLAPPED\_DOWN\_SAMPLE\_RATIO = 59  
SAMPLE\_PROPERTIES\_OVERLAPPED\_DOWN\_SAMPLE\_RATIO\_MODE = 60  
SAMPLE\_PROPERTIES\_OVERLAPPED\_REQUERSTED\_NO\_OF\_SAMPLES = 61  
SAMPLE\_PROPERTIES\_OVERLAPPED\_SEGMENT\_INDEX\_FROM = 62  
SAMPLE\_PROPERTIES\_OVERLAPPED\_SEGMENT\_INDEX\_TO = 63  
SIGNAL\_GENERATOR = 64  
SIGNAL\_GENERATOR\_BUILT\_IN = 65  
SIGNAL\_GENERATOR\_BUILT\_IN\_WAVE\_TYPE = 66  
SIGNAL\_GENERATOR\_BUILT\_IN\_START\_FREQUENCY = 67  
SIGNAL\_GENERATOR\_BUILT\_IN\_STOP\_FREQUENCY = 68  
SIGNAL\_GENERATOR\_BUILT\_IN\_INCREMENT = 69  
SIGNAL\_GENERATOR\_BUILT\_IN\_DWELL\_TIME = 70  
SIGNAL\_GENERATOR\_AWG = 71  
SIGNAL\_GENERATOR\_AWG\_START\_DELTA\_PHASE = 72  
SIGNAL\_GENERATOR\_AWG\_STOP\_DELTA\_PHASE = 73  
SIGNAL\_GENERATOR\_AWG\_DELTA\_PHASE\_INCREMENT = 74  
SIGNAL\_GENERATOR\_AWG\_DWELL\_COUNT = 75  
SIGNAL\_GENERATOR\_AWG\_INDEX\_MODE = 76  
SIGNAL\_GENERATOR\_AWG\_WAVEFORM\_SIZE = 77  
SIGNAL\_GENERATOR\_ARRAY\_OF\_AWG\_WAVEFORM\_VALUES = 78  
SIGNAL\_GENERATOR\_OFFSET\_VOLTAGE = 79  
SIGNAL\_GENERATOR\_PK\_TO\_PK = 80  
SIGNAL\_GENERATOR\_OPERATION = 81  
SIGNAL\_GENERATOR\_SHOTS = 82  
SIGNAL\_GENERATOR\_SWEEPS = 83  
SIGNAL\_GENERATOR\_SWEEP\_TYPE = 84  
SIGNAL\_GENERATOR\_TRIGGER\_TYPE = 85  
SIGNAL\_GENERATOR\_TRIGGER\_SOURCE = 86

**SIGNAL\_GENERATOR\_EXT\_IN\_THRESHOLD = 87**

**ETS = 88**

**ETS\_STATE = 89**

**ETS\_CYCLE = 90**

**ETS\_INTERLEAVE = 91**

**ETS\_SAMPLE\_TIME\_PICOSECONDS = 92**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000Channel`  
Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**AUX = 5**

**MAX\_TRIGGER\_SOURCES = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex`  
Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**MAX = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000Range`  
Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**

**R\_20MV = 1**

**R\_50MV = 2**

**R\_100MV = 3**

**R\_200MV = 4**

**R\_500MV = 5**

**R\_1V = 6**

**R\_2V = 7**

**R\_5V = 8**

**R\_10V = 9**

**R\_20V = 10**

**R\_50V = 11**

**R\_MAX = 12**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000EtsMode`

Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000TimeUnits`

Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000SweepType`

Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType`

Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**

**RAMP\_UP = 3**

**RAMP\_DOWN = 4**

**SINC = 5**

**GAUSSIAN = 6**

**HALF\_SINE = 7**

**DC\_VOLTAGE = 8**

**WHITE\_NOISE = 9**

**MAX = 10**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000SigGenTrigType`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH = 2**

**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000SigGenTrigSource`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**SCOPE\_TRIG = 1**

**AUX\_IN = 2**

**EXT\_IN = 3**

**SOFT\_TRIG = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000IndexMode`

Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**

**DUAL = 1**

**QUAD = 2**

**MAX = 3**



**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdDirection`

Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000TriggerState`

Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000RatioMode`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**DECIMATE = 2**

**AVERAGE = 4**

**DISTRIBUTION = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000PulseWidthType`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelInfo`  
Bases: `enum.IntEnum`

An enumeration.

**RANGES = 0**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ADeviceResolution`  
Bases: `enum.IntEnum`

An enumeration.

**RES\_8BIT = 0**

**RES\_12BIT = 1**

**RES\_14BIT = 2**

**RES\_15BIT = 3**

**RES\_16BIT = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AExtraOperations`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**WHITENOISE = 1**

**PRBS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ABandwidthLimiter`  
Bases: `enum.IntEnum`

An enumeration.

**BW\_FULL = 0**

**BW\_20MHZ = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ACoupling`  
Bases: `enum.IntEnum`

An enumeration.

**AC = 0**

**DC = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel`

Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**AUX = 5**

**MAX\_TRIGGER\_SOURCES = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex`

Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**MAX = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ARange`

Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**

**R\_20MV = 1**

**R\_50MV = 2**

**R\_100MV = 3**

**R\_200MV = 4**

**R\_500MV = 5**

**R\_1V = 6**

**R\_2V = 7**

**R\_5V = 8**

**R\_10V = 9**

**R\_20V = 10**

**R\_50V = 11**

**R\_MAX = 12**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AetsMode`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ATimeUnits`  
Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ASweepType`  
Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType`  
Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**

**RAMP\_UP = 3**

**RAMP\_DOWN = 4**

**SINC = 5**

**GAUSSIAN = 6**

**HALF\_SINE = 7**

**DC\_VOLTAGE = 8**

**WHITE\_NOISE = 9**

**MAX = 10**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ASigGenTrigType`

Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH = 2**

**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ASigGenTrigSource`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**SCOPE\_TRIG = 1**

**AUX\_IN = 2**

**EXT\_IN = 3**

**SOFT\_TRIG = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AIndexMode`

Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**

**DUAL = 1**

**QUAD = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdMode`

Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection`  
Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**ABOVE\_LOWER = 5**

**BELOW\_LOWER = 6**

**RISING\_LOWER = 7**

**FALLING\_LOWER = 8**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**POSITIVE\_RUNT = 9**

**NEGATIVE\_RUNT = 10**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ATriggerState`  
Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ATriggerWithinPreTrig`  
Bases: `enum.IntEnum`

An enumeration.

**DISABLE = 0**

**ARM = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**DECIMATE = 2**

**AVERAGE = 4**

**DISTRIBUTION = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000APulseWidthType`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelInfo`  
Bases: `enum.IntEnum`

An enumeration.

**RANGES = 0**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency`  
Bases: `enum.IntEnum`

An enumeration.

**EF\_OFF = 0**

**EF\_5MHZ = 1**

**EF\_10MHZ = 2**

**EF\_20MHZ = 3**

**EF\_25MHZ = 4**

**EF\_MAX = 5**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000BandwidthLimiter`  
Bases: `enum.IntEnum`

An enumeration.

**BW\_FULL = 0**

**BW\_20MHZ = 1**

**BW\_25MHZ = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000Channel`  
Bases: `enum.IntEnum`

An enumeration.

**A = 0**

**B = 1**

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**AUX = 5**

**MAX\_TRIGGER\_SOURCES = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex`  
Bases: `enum.IntEnum`

An enumeration.

**A\_MAX = 0**

**A\_MIN = 1**

**B\_MAX = 2**

**B\_MIN = 3**

**C\_MAX = 4**

**C\_MIN = 5**

**D\_MAX = 6**

**D\_MIN = 7**

**MAX = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000Range`  
Bases: `enum.IntEnum`

An enumeration.

**R\_10MV = 0**

**R\_20MV = 1**

**R\_50MV = 2**

**R\_100MV = 3**

**R\_200MV = 4**

**R\_500MV = 5**

**R\_1V = 6**

**R\_2V = 7**

**R\_5V = 8**

**R\_10V = 9**

**R\_20V = 10**

**R\_50V = 11**

**R\_MAX = 12**



**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000Coupling`  
Bases: `enum.IntEnum`

An enumeration.

**AC = 0**

**DC\_1M = 1**

**DC\_50R = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000EtsMode`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000TimeUnits`  
Bases: `enum.IntEnum`

An enumeration.

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000SweepType`  
Bases: `enum.IntEnum`

An enumeration.

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000WaveType`  
Bases: `enum.IntEnum`

An enumeration.

**SINE = 0**

**SQUARE = 1**

**TRIANGLE = 2**  
**RAMP\_UP = 3**  
**RAMP\_DOWN = 4**  
**SINC = 5**  
**GAUSSIAN = 6**  
**HALF\_SINE = 7**  
**DC\_VOLTAGE = 8**  
**MAX = 9**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000ExtraOperations`  
Bases: `enum.IntEnum`

An enumeration.

**OFF = 0**  
**WHITENOISE = 1**  
**PRBS = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000SigGenTrigType`  
Bases: `enum.IntEnum`

An enumeration.

**RISING = 0**  
**FALLING = 1**  
**GATE\_HIGH = 2**  
**GATE\_LOW = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000SigGenTrigSource`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**  
**SCOPE\_TRIG = 1**  
**AUX\_IN = 2**  
**EXT\_IN = 3**  
**SOFT\_TRIG = 4**  
**TRIGGER\_RAW = 5**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000IndexMode`  
Bases: `enum.IntEnum`

An enumeration.

**SINGLE = 0**  
**DUAL = 1**  
**QUAD = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdMode`  
Bases: `enum.IntEnum`

An enumeration.

**LEVEL = 0**

**WINDOW = 1**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdDirection`  
Bases: `enum.IntEnum`

An enumeration.

**ABOVE = 0**

**BELOW = 1**

**RISING = 2**

**FALLING = 3**

**RISING\_OR\_FALLING = 4**

**ABOVE\_LOWER = 5**

**BELOW\_LOWER = 6**

**RISING\_LOWER = 7**

**FALLING\_LOWER = 8**

**INSIDE = 0**

**OUTSIDE = 1**

**ENTER = 2**

**EXIT = 3**

**ENTER\_OR\_EXIT = 4**

**POSITIVE\_RUNT = 9**

**NEGATIVE\_RUNT = 10**

**NONE = 2**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000TriggerState`  
Bases: `enum.IntEnum`

An enumeration.

**DONT\_CARE = 0**

**TRUE = 1**

**FALSE = 2**

**MAX = 3**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000RatioMode`  
Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**AGGREGATE = 1**

**AVERAGE = 2**

**DECIMATE = 4**

**DISTRIBUTION = 8**

**class** `msl.equipment.resources.picotech.picoscope.enums.PS6000PulseWidthType`

Bases: `enum.IntEnum`

An enumeration.

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

### **`msl.equipment.resources.picotech.picoscope.errors` module**

Exceptions and error codes defined in the Pico Technology SDK v10.6.10.24

### **`msl.equipment.resources.picotech.picoscope.functions` module**

Functions defined in the Pico Technology SDK v10.6.10.24

### **`msl.equipment.resources.picotech.picoscope.helper` module**

The functions in this module are only helper functions that were initially used for wrapping the PicoScope SDK in Python. There are no user-facing functions here, only those used by a developer.

These functions are used to

Print the following to stdout

- the `#define` constants
- the function signatures for the PicoScope subclasses
- functions with similar function signatures

Create the following files

- `picoscope_enums.py`
- `picoscope_structs.py`
- `picoscope_callbacks.py`
- `picoscope_function_pointers.py`

`msl.equipment.resources.picotech.picoscope.helper.parse_pico_scope_api_header` (*partially shown*)  
Parse a PicoScope header file.

**Args:** path (str): The path to a PicoScope header file

**Returns:** dict: {'enums': {}, 'defines': {}, 'functions': {}, 'structs': {}, 'functypes': {}}

`msl.equipment.resources.picotech.picoscope.helper.print_define_statements (header_`  
 Print the #define constants in the PicoScope header files to stdout

The output is copied and pasted to the appropriate PicoScope subclass.

For example, the stdout text below

```
ps5000aApi
```

is copied to

```
class PicoScope5000A(PicoScope):
```

`msl.equipment.resources.picotech.picoscope.helper.create_picoscope_enums_file (hea`  
 Creates the `_picoscope_enums.py` file

`msl.equipment.resources.picotech.picoscope.helper.create_picoscope_structs_file (`  
 Creates the `_picoscope_structs.py` file

`msl.equipment.resources.picotech.picoscope.helper.check_enum_struct_names ()`  
 Ensure that none of the items in `ENUM_DATA_TYPE_NAMES` are in  
`STRUCT_DATA_TYPE_ALIASES`

`msl.equipment.resources.picotech.picoscope.helper.create_callbacks_file (header_dict`  
 Create the `_picoscope_callbacks.py` file

`msl.equipment.resources.picotech.picoscope.helper.ctypes_map (dtype,`  
`hkey)`

`msl.equipment.resources.picotech.picoscope.helper.create_picoscope_functions_fil`  
 Create the `_picoscope_functions.py` file. The lists in this file are used for creating ctypes.\_FuncPtr  
 objects

`msl.equipment.resources.picotech.picoscope.helper.print_class_def_signatures (head`

`msl.equipment.resources.picotech.picoscope.helper.print_common_functions (header_di`  
`ps_name)`

### **msl.equipment.resources.picotech.picoscope.picoscope module**

Base class for a PicoScope from Pico Technology.

`msl.equipment.resources.picotech.picoscope.picoscope.enumerate_units ()`  
 Find the PicoScopes that are connected to the computer.

This function counts the number of PicoScopes connected to the computer, and returns a list of  
 serial numbers as a string.

---

**Note:** It seems as though you cannot call this function after you have opened a connection to a  
 PicoScope.

---

**Returns** `list` of `str` – A list of serial numbers of the PicoScopes that were found.

**class** `msl.equipment.resources.picotech.picoscope.picoscope.PicoScope` (`record`,  
`func_ptrs`)

Bases: `msl.equipment.connection_msl.ConnectionSDK`

Use the PicoScope SDK to communicate with the oscilloscope.

Do not instantiate this class directly. Use the factory method, `msl.equipment.factory.connect`, or the `record` object itself, `record.connect()`, to connect to the equipment.

The `record.connection.properties` dictionary for a PicoScope supports the following key-value pairs:

```
'open_unit': bool, # default is True
'open_unit_async': bool, # default is False
'auto_select_power': bool # for PicoScopes that can be powered by an_
↪AC adaptor or a USB cable, default is True
'resolution': '14bit', # only valid for ps5000a, default is '8bit'
```

The SDK version that was initially used to create this base class and the PicoScope subclasses was *Pico Technology SDK 64-bit v10.6.10.24*

#### Parameters

- **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.
- **func\_ptrs** (*functions*) – The appropriate function-pointer list for the SDK.

#### handle

Returns the handle to the SDK library.

#### channel

`dict` of `channel.PicoScopeChannel` – The information about each channel

#### dt

`float` – The time between voltage samples (i.e., delta t).

#### pre\_trigger

`float` – The number of seconds that data was acquired for before the trigger event.

#### close\_unit()

Disconnect from the PicoScope.

#### disconnect()

Disconnect from the PicoScope.

#### get\_unit\_info (info=None, include\_name=True)

Retrieves information about the PicoScope.

This function retrieves information about the specified oscilloscope. If the device fails to open, or no device is opened only the driver version is available.

#### Parameters

- **info** (*PicoScopeInfo*, optional) – An enum value, or if `None` then request all information from the PicoScope.

- **include\_name** (`bool`, optional) – If `True` then includes the enum member name as a prefix. For example, return `CAL_DATE: 09Aug16` if `True` else `09Aug16`.

**Returns** `str` – The requested information from the PicoScope.

### **is\_ready()**

Has the PicoScope collecting the requested number of samples?

This function may be used instead of a callback function to receive data from `run_block()`. To use this method, pass `None` as the callback parameter in `run_block()`. You must then poll the driver to see if it has finished collecting the requested samples.

**Returns** `bool` – Whether the PicoScope has collected the requested number of samples.

### **maximum\_value()**

`int`: This function returns the maximum ADC count.

### **minimum\_value()**

`int`: This function returns the minimum ADC count.

### **ping\_unit()**

Ping the PicoScope.

This function can be used to check that the already opened device is still connected to the USB port and communication is successful.

### **run\_block(pre\_trigger=0.0, callback=None, segment\_index=0)**

Start collecting data in block mode.

All input arguments are ignored for ps2000 and ps3000.

#### **Parameters**

- **pre\_trigger** (`float`) – The number of seconds before the trigger event to start acquiring data.
- **segment\_index** (`int`) – Specifies which memory segment to save the data to (see manual).
- **callback** (`callbacks.BlockReady`) – A `BlockReady` callback function.

### **run\_streaming(pre\_trigger=0.0, auto\_stop=True, factor=1, ratio\_mode='NONE')**

Start collecting data in streaming mode.

This function tells the oscilloscope to start collecting data in streaming mode. When data has been collected from the device it is down sampled if necessary and then delivered to the application. Call `get_streaming_latest_values()` to retrieve the data.

When a trigger is set, the total number of samples stored in the driver is the sum of `max_pre_trigger_samples` and `max_post_trigger_samples`. If `auto_stop` is false then this will become the maximum number of samples without down sampling.

The `ratio_mode` argument is ignored for ps4000 and ps5000.

#### **Parameters**

- **pre\_trigger** (`float`) – The number of seconds before the trigger event to start acquiring data.
- **auto\_stop** (`bool`) – A flag that specifies if the streaming should stop when all of samples have been captured.
- **factor** (`int`) – The down-sampling factor that will be applied to the raw data.
- **ratio\_mode** (`enum.IntEnum`) – Which down-sampling mode to use.

**set\_channel** (*channel*, *coupling='dc'*, *scale='10V'*, *offset=0.0*, *bandwidth='full'*, *enabled=True*)  
Configure a channel.

This function specifies whether an input channel is to be enabled, its input coupling type, voltage range, analog offset and bandwidth limit. Some of the arguments within this function have model-specific values. Please consult the manual according to the model you have.

The *bandwidth* argument is only used for ps6000.

The *offset* and *bandwidth* arguments are ignored for ps2000, ps3000, ps4000 and ps5000.

#### Parameters

- **channel** (`enum.IntEnum`) – The channel to be configured
- **coupling** (`enum.IntEnum`) – The impedance and coupling type.
- **scale** (`enum.IntEnum`) – The input voltage range.
- **offset** (`float`) – A voltage to add to the input channel before digitization. The allowable range of offsets depends on the input range selected for the channel, as obtained from `get_analogue_offset()`.
- **bandwidth** (`enum.IntEnum`) – The bandwidth limiter to use.
- **enabled** (`bool`) – Whether to enable the channel.

**set\_timebase** (*dt*, *duration*, *segment\_index=0*, *oversample=0*)  
Set the timebase information.

The *segment\_index* is ignored for ps2000 and ps3000.

The *oversample* argument is ignored by ps2000a, ps3000a, ps4000a and ps5000a.

#### Parameters

- **dt** (`float`) – The sampling interval, in seconds.
- **duration** (`float`) – The number of seconds to acquire data for.
- **segment\_index** (`int`) – Which memory segment to save the data to.
- **oversample** (`int`) – The amount of over-sample required.

**Raises** `ConnectionError` – If the timebase or duration is invalid.

**set\_trigger** (*channel*, *threshold*, *delay=0.0*, *direction='rising'*, *timeout=0.1*, *enable=True*)  
Set up the trigger.

#### Parameters

- **channel** (`enum.IntEnum`) – The trigger channel.



- **threshold** (`float`) – The threshold voltage to signal a trigger event.
- **delay** (`float`) – The time, in seconds, between the trigger occurring and the first sample.
- **direction** (`enum.IntEnum`) – The direction in which the signal must move to cause a trigger.
- **timeout** (`float`) – The time, in seconds, to wait to automatically create a trigger event if no trigger event occurs. If `timeout <= 0` then wait indefinitely for a trigger. Only accurate to the nearest millisecond.
- **enable** (`bool`) – Set to `False` to disable the trigger for this channel. Not used for ps2000 or ps3000.

**stop ()**

Stop the oscilloscope from sampling data.

If this function is called before a trigger event occurs, then the oscilloscope may not contain valid data.

**wait\_until\_ready ()**

Blocking function to wait for the scope to finish acquiring data.

**set\_pulse\_width\_qualifier** (`conditions`, `direction`, `lower`, `upper`, `pulse_width_type`)

This function sets up pulse width qualification, which can be used on its own for pulse width triggering or combined with other triggering to produce more complex triggers. The pulse width qualifier is set by defining a list of `PwqConditions`` structures.

**msl.equipment.resources.picotech.picoscope.picoscope\_2k3k module**

Base class for the ps2000 and ps3000 PicoScopes.

**class** `msl.equipment.resources.picotech.picoscope.picoscope_2k3k.PicoScope2k3k` (`record`, `func_ptr`)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope.PicoScope`

Use the PicoScope SDK to communicate with ps2000 and ps3000 oscilloscopes.

Do not instantiate this class directly. Use the factory method, `msl.equipment.factory.connect`, or the `record` object itself, `record.connect ()`, to connect to the equipment.

**Parameters**

- **record** (`EquipmentRecord`) – An equipment record from an **Equipment-Register Database**.
- **func\_ptrs** (`functions`) – The appropriate function-pointer list for the SDK.

**errcheck\_zero** (`result`, `func`, `args`)

If the SDK function returns 0 then raise an exception.

**errcheck\_one** (`result`, `func`, `args`)

If the SDK function returns 1 then raise an exception.

**errcheck\_negative\_one** (*result, func, args*)

If the SDK function returns -1 then raise an exception.

**flash\_led** ()

Flashes the LED on the front of the oscilloscope three times and returns within one second.

**get\_streaming\_last\_values** (*lp\_get\_overview\_buffers\_max\_min*)

This function is used to collect the next block of values while fast streaming is running. You must call `run_streaming_ns()` beforehand to set up fast streaming.

**get\_streaming\_values** (*no\_of\_values, no\_of\_samples\_per\_aggregate*)

This function is used after the driver has finished collecting data in fast streaming mode. It allows you to retrieve data with different aggregation ratios, and thus zoom in to and out of any region of the data.

**get\_streaming\_values\_no\_aggregation** (*no\_of\_values*)

This function retrieves raw streaming data from the driver's data store after fast streaming has stopped.

**get\_timebase** (*timebase, no\_of\_samples, oversample=0*)

This function discovers which timebases are available on the oscilloscope. You should set up the channels using `set_channel()` and, if required, ETS mode using `set_ets()` first. Then call this function with increasing values of timebase, starting from 0, until you find a timebase with a sampling interval and sample count close enough to your requirements.

**get\_times\_and\_values** (*time\_units, no\_of\_values*)

This function is used to get values and times in block mode after calling `run_block()`.

**get\_values** (*num\_values*)

This function is used to get values in compatible streaming mode after calling `run_streaming()`, or in block mode after calling `run_block()`.

**open\_unit** ()

This function opens a PicoScope 2000/3000 Series oscilloscope. The driver can support up to 64 oscilloscopes.

**open\_unit\_async** ()

This function opens a PicoScope 2000/3000 Series oscilloscope without waiting for the operation to finish. You can find out when it has finished by periodically calling `open_unit_progress()`, which returns a value of 100 when the scope is open.

The driver can support up to 64 oscilloscopes.

**open\_unit\_progress** ()

This function checks on the progress of `open_unit_async()`.

The function will return a value from 0 to 100, where 100 implies that the operation is complete.

**overview\_buffer\_status** ()

This function indicates whether or not the overview buffers used by `run_streaming_ns()` have overrun. If an overrun occurs, you can choose to increase the `overview_buffer_size` argument that you pass in the next call to `run_streaming_ns()`.

**set\_adv\_trigger\_channel\_directions** (*channel\_a, channel\_b, channel\_c, channel\_d, ext*)

This function sets the direction of the trigger for each channel.

**set\_adv\_trigger\_delay** (*delay, pre\_trigger\_delay*)

This function sets the pre-trigger and post-trigger delays. The default action, when both these delays are zero, is to start capturing data beginning with the trigger event and to stop a specified time later. The start of capture can be delayed by using a nonzero value of *delay*. Alternatively, the start of capture can be advanced to a time before the trigger event by using a negative value of *pre\_trigger\_delay*. If both arguments are non-zero then their effects are added together.

**set\_ets** (*mode, ets\_cycles, ets\_interleave*)

This function is used to enable or disable ETS (equivalent time sampling) and to set the ETS parameters.

**set\_trigger** (*source, threshold, direction, delay, auto\_trigger\_ms*)

This function just calls `set_trigger2()`, since Python supports a floating-point value for defining the *delay* parameter.

**set\_trigger2** (*source, threshold, direction, delay, auto\_trigger\_ms*)

This function is used to enable or disable triggering and its parameters. It has the same behaviour as `set_trigger()`, except that the *delay* parameter is a floating-point value.

For oscilloscopes that support advanced triggering, see `set_adv_trigger_channel_conditions()` and related functions.

**set\_adv\_trigger\_channel\_properties** (*channel\_properties, auto\_trigger\_milliseconds*)

This function is used to enable or disable triggering and set its parameters.

**set\_adv\_trigger\_channel\_conditions** (*conditions*)

This function sets up trigger conditions on the scope's inputs. The trigger is set up by defining a list of `picoscope_structs TriggerConditions` structures. Each structure is the AND of the states of one scope input.

**msl.equipment.resources.picotech.picoscope.picoscope\_api module**

Base class for the PicoScopes that have a header file which ends with `*Api.h`.

Namely, `ps2000aApi`, `ps3000aApi`, `ps4000aApi`, `ps4000aApi`, `ps5000aApi`, `ps5000aApi` and `ps6000aApi`.

**class** `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi` (*record, func\_ptrs*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope.PicoScope`

Base class for the PicoScopes that have a header file which ends with `*Api.h`.

Use the PicoScope SDK to communicate with the `ps2000a`, `ps3000a`, `ps4000`, `ps4000a`, `ps5000`, `ps5000a` and `ps6000` oscilloscopes.

Do not instantiate this class directly. Use the factory method, `msl.equipment.factory.connect`, or the *record* object itself, `record.connect()`, to connect to the equipment.

**Parameters**

- **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.
- **func\_ptrs** (*functions*) – The appropriate function-pointer list for the SDK.

**errcheck\_api** (*result, func, args*)

The SDK function returns PICO\_OK if the function call was successful.

**change\_power\_source** (*power\_state*)

Change the power source.

This function selects the power supply mode. You must call this function if any of the following conditions arises:

- USB power is required
- the AC power adapter is connected or disconnected during use
- a USB 3.0 scope is plugged into a USB 2.0 port (indicated if any function returns the PICO\_USB3\_0\_DEVICE\_NON\_USB3\_0\_PORT status code)

This function is only valid for ps3000a, ps4000a and ps5000a.

**current\_power\_source** ()

This function returns the current power state of the device.

This function is only valid for ps3000a, ps4000a and ps5000a.

**flash\_led** (*action*)

This function flashes the LED on the front of the scope without blocking the calling thread. Calls to `run_streaming()` and `run_block()` cancel any flashing started by this function. It is not possible to set the LED to be constantly illuminated, as this state is used to indicate that the scope has not been initialized.

**get\_analogue\_offset** (*voltage\_range, coupling*)

This function is used to get the maximum and minimum allowable analog offset for a specific voltage range.

This function is invalid for ps4000 and ps5000.

**get\_channel\_information** (*channel, info='ranges'*)

This function queries which ranges are available on a scope device.

This function is invalid for ps5000 and ps6000.

#### Parameters

- **channel** (`enum.IntEnum`) – 0=ChannelA, 1=ChannelB, ...
- **info** (`enum.IntEnum`) – A `ChannelInfo` enum value or enum member name.

**get\_max\_down\_sample\_ratio** (*num\_unaggregated\_samples, mode='None', segment\_index=0*)

**Returns** `int` – This function returns the maximum down-sampling ratio that can be used for a given number of samples in a given down-sampling mode.

**get\_max\_segments** ()

This function is valid for ps2000a, ps3000a, ps4000a and ps5000a.

**Returns** `int` – This function returns the maximum number of segments allowed for the opened device. This number is the maximum value of `nsegments` that can be passed to `memory_segments()`.

**get\_no\_of\_captures** ()

This function finds out how many captures are available in rapid block mode after

`run_block()` has been called when either the collection completed or the collection of waveforms was interrupted by calling `stop()`. The returned value (`nCaptures`) can then be used to iterate through the number of segments using `get_values()`, or in a single call to `get_values_bulk()` where it is used to calculate the `toSegmentIndex` parameter.

Not valid for ps5000.

#### **get\_num\_of\_processed\_captures()**

This function finds out how many captures in rapid block mode have been processed after `run_block()` has been called when either the collection completed or the collection of waveforms was interrupted by calling `stop()`. The returned value (`nCaptures`) can then be used to iterate through the number of segments using `get_values()`, or in a single call to `get_values_bulk()` where it is used to calculate the `toSegmentIndex` parameter.

Not valid for ps4000 and ps5000.

#### **get\_streaming\_latest\_values(lp\_ps)**

This function instructs the driver to return the next block of values to your `streaming_ready()` callback. You must have previously called `run_streaming()` beforehand to set up streaming.

#### **get\_timebase(timebase, num\_samples=0, segment\_index=0, oversample=0)**

Since Python supports the `float` data type, this function returns `get_timebase2()`. The timebase that is returned is in **seconds** (not ns).

This function calculates the sampling rate and maximum number of samples for a given timebase under the specified conditions. The result will depend on the number of channels enabled by the last call to `set_channel()`.

The *oversample* argument is only used by ps4000, ps5000 and ps6000.

#### **get\_timebase2(timebase, num\_samples=0, segment\_index=0, oversample=0)**

This function is an upgraded version of `get_timebase()`, and returns the time interval as a `float` rather than an `int`. This allows it to return sub-nanosecond time intervals. See `get_timebase()` for a full description.

The timebase that is returned is in **seconds** (not ns).

The *oversample* argument is only used by ps4000, ps5000 and ps6000.

#### **get\_trigger\_time\_offset(segment\_index=0)**

This function gets the time, as two 4-byte values, at which the trigger occurred. Call it after block-mode data has been captured or when data has been retrieved from a previous block-mode capture. A 64-bit version of this function, `get_trigger_time_offset64()`, is also available.

#### **get\_trigger\_time\_offset64(segment\_index=0)**

This function gets the time, as a single 64-bit value, at which the trigger occurred. Call it after block-mode data has been captured or when data has been retrieved from a previous block-mode capture. A 32-bit version of this function, `get_trigger_time_offset()`, is also available.

#### **get\_values(num\_samples=None, start\_index=0, factor=1, ratio\_mode='None', segment\_index=0)**

This function returns block-mode data, with or without down sampling, starting at the specified sample number. It is used to get the stored data from the driver after data collection has stopped.

### Parameters

- **num\_samples** (`int` or `None`) – The number of samples required. If `None` then automatically determine the number of samples to retrieve.
- **start\_index** (`int`) – A zero-based index that indicates the start point for data collection. It is measured in sample intervals from the start of the buffer.
- **factor** (`int`) – The down-sampling factor that will be applied to the raw data.
- **ratio\_mode** (`enum.IntEnum`) – Which down-sampling mode to use. A `RatioMode` enum.
- **segment\_index** (`int`) – The zero-based number of the memory segment where the data is stored.

**get\_values\_async** (*lp\_data\_ready*, *num\_samples=None*, *start\_index=0*, *factor=1*, *ratio\_mode='None'*, *segment\_index=0*)

This function returns data either with or without down sampling, starting at the specified sample number. It is used to get the stored data from the scope after data collection has stopped. It returns the data using the `lp_data_ready` callback.

### Parameters

- **lp\_data\_ready** (*ctypes callback function*) – A `DataReady` *callback* function.
- **num\_samples** (`int` or `None`) – The number of samples required. If `None` then automatically determine the number of samples to retrieve.
- **start\_index** (`int`) – A zero-based index that indicates the start point for data collection. It is measured in sample intervals from the start of the buffer.
- **factor** (`int`) – The downsampling factor that will be applied to the raw data.
- **ratio\_mode** (`enum.IntEnum`) – Which down-sampling mode to use. A `RatioMode` enum.
- **segment\_index** (`int`) – The zero-based number of the memory segment where the data is stored.

**get\_values\_bulk** (*from\_segment\_index=0*, *to\_segment\_index=None*, *factor=1*, *ratio\_mode='None'*)

This function retrieves waveforms captured using rapid block mode. The waveforms must have been collected sequentially and in the same run.

The `down_sample_ratio` and `down_sample_ratio_mode` arguments are ignored for ps4000 and ps5000.

**get\_values\_overlapped** (*start\_index*, *down\_sample\_ratio*, *down\_sample\_ratio\_mode*, *segment\_index*)

This function allows you to make a deferred data-collection request, which will later be executed, and the arguments validated, when you call `run_block()` in block mode. The advantage of this function is that the driver makes contact with the scope only once, when you call `run_block()`, compared with the two contacts that occur when you use the

conventional `run_block()`, `get_values()` calling sequence. This slightly reduces the dead time between successive captures in block mode.

This function is invalid for ps4000 and ps5000.

**get\_values\_overlapped\_bulk** (*start\_index*, *down\_sample\_ratio*,  
*down\_sample\_ratio\_mode*, *from\_segment\_index*,  
*to\_segment\_index*)

This function allows you to make a deferred data-collection request, which will later be executed, and the arguments validated, when you call `run_block()` in rapid block mode. The advantage of this method is that the driver makes contact with the scope only once, when you call `run_block()`, compared with the two contacts that occur when you use the conventional `run_block()`, `get_values_bulk()` calling sequence. This slightly reduces the dead time between successive captures in rapid block mode.

This function is invalid for ps4000 and ps5000.

**get\_values\_trigger\_time\_offset\_bulk** (*from\_segment\_index*,  
*to\_segment\_index*)

This function retrieves the time offsets, as lower and upper 32-bit values, for waveforms obtained in rapid block mode. This function is provided for use in programming environments that do not support 64-bit integers. If your programming environment supports this data type, it is easier to use `get_values_trigger_time_offset_bulk64()`.

**get\_values\_trigger\_time\_offset\_bulk64** (*from\_segment\_index=0*,  
*to\_segment\_index=None*)

This function retrieves the 64-bit time offsets for waveforms captured in rapid block mode.

A 32-bit version of this function, `get_values_trigger_time_offset_bulk()`, is available for use with programming languages that do not support 64-bit integers

**hold\_off** (*holdoff*, *holdoff\_type*)

This function is for backward compatibility only and is not currently used.

This function is only defined for ps2000a, ps3000a and ps4000.

**is\_led\_flashing** ()

This function reports whether or not the LED is flashing.

This function is supported by ps4000, ps4000a and ps5000.

**is\_trigger\_or\_pulse\_width\_qualifier\_enabled** ()

This function discovers whether a trigger, or pulse width triggering, is enabled.

**memory\_segments** (*num\_segments*)

This function sets the number of memory segments that the scope will use. When the scope is opened, the number of segments defaults to 1, meaning that each capture fills the scopes available memory. This function allows you to divide the memory into a number of segments so that the scope can store several waveforms sequentially.

**no\_of\_streaming\_values** ()

This function returns the number of samples available after data collection in streaming mode. Call it after calling `stop()`.

**open\_unit** (*auto\_select\_power=True*, *resolution='8Bit'*)

Open the PicoScope for communication.

This function opens a PicoScope attached to the computer. The maximum number of units that can be opened depends on the operating system, the kernel driver and the computer.

### Parameters

- **auto\_select\_power** (*bool*, optional) – PicoScopes that can be powered by either DC power or by USB power may raise `PICO_POWER_SUPPLY_NOT_CONNECTED` if the DC power supply is not connected. Passing in `True` will automatically switch to the USB power source.
- **resolution** (*str*, optional) – The ADC resolution: 8, 12, 14, 15 or 16Bit. Only used by the PS5000A Series and it is ignored for all other PicoScope Series.

**open\_unit\_async** (*auto\_select\_power=True, resolution='8Bit'*)

This function opens a scope without blocking the calling thread. You can find out when it has finished by periodically calling `open_unit_progress()` until that function returns a value of 100.

### Parameters

- **auto\_select\_power** (*bool*, optional) – PicoScopes that can be powered by either DC power or by USB power may raise `PICO_POWER_SUPPLY_NOT_CONNECTED` if the DC power supply is not connected. Passing in `True` will automatically switch to the USB power source.
- **resolution** (*str*, optional) – The ADC resolution: 8, 12, 14, 15 or 16Bit. Only used by the PS5000A Series and it is ignored for all other PicoScope Series.

**open\_unit\_progress** ()

This function checks on the progress of a request made to `open_unit_async()` to open a scope. The return value is from 0 to 100, where 100 implies that the operation is complete.

**set\_bandwidth\_filter** (*channel, bandwidth*)

This function is reserved for future use.

This function is only valid for ps3000a, ps4000a and ps5000a.

**set\_data\_buffer** (*channel, buffer=None, mode='None', segment\_index=0*)

Set the data buffer for the specified channel.

The *mode* argument is ignored for ps4000 and ps5000. The *segment\_index* argument is ignored for ps4000, ps5000 and ps6000.

### Parameters

- **channel** (*enum.IntEnum*) – An enum value or member name from `Channel`.
- **buffer** (*numpy.ndarray* or `None`) – A int16, numpy array. If `None` then use a pre-allocated array.
- **mode** (*enum.IntEnum*) – An enum value or member name from `RatioMode`.
- **segment\_index** (*int*) – The zero-based number of the memory segment where the data is stored.

**set\_data\_buffer\_bulk** (*channel, buffer, waveform, mode='None'*)

This function allows you to associate a buffer with a specified waveform number and input



channel in rapid block mode. The number of waveforms captured is determined by the `nCaptures` argument sent to `set_no_of_captures()`. There is only one buffer for each waveform because the only down-sampling mode that requires two buffers, aggregation mode, is not available in rapid block mode. Call one of the `GetValues` functions to retrieve the data after capturing.

This function is only valid for ps4000, ps5000 and ps6000.

The `down_sample_ratio_mode` argument is ignored for ps4000 and ps5000.

**set\_data\_buffers** (*channel, buffer\_length, mode, segment\_index*)

This function tells the driver where to store the data, either unprocessed or down sampled, that will be returned after the next call to one of the `GetValues` functions. The function allows you to specify only a single buffer, so for aggregation mode, which requires two buffers, you need to call `set_data_buffers()` instead.

You must allocate memory for the buffer before calling this function.

The `mode` argument is ignored for ps4000 and ps5000.

The `segment_index` argument is ignored for ps4000, ps5000 and ps6000.

**set\_digital\_port** (*port, enabled, logic\_level*)

This function is used to enable the digital port and set the logic level (the voltage at which the state transitions from 0 to 1).

This function is only used by ps2000a and ps3000a.

**set\_ets** (*mode, ets\_cycles, ets\_interleave*)

This function is used to enable or disable ETS (equivalent-time sampling) and to set the ETS parameters. See ETS overview for an explanation of ETS mode.

**set\_ets\_time\_buffer** (*buffer*)

This function tells the driver where to find your applications ETS time buffers. These buffers contain the 64-bit timing information for each ETS sample after you run a block-mode ETS capture.

**Parameters** `buffer` (*ctypes array of c\_int64*) – An array of 64-bit words, each representing the time in picoseconds at which the sample was captured.

**set\_ets\_time\_buffers** (*time\_upper, time\_lower*)

This function tells the driver where to find your applications ETS time buffers. These buffers contain the timing information for each ETS sample after you run a blockmode ETS capture. There are two buffers containing the upper and lower 32-bit parts of the timing information, to allow programming languages that do not support 64-bit data to retrieve the timings.

**set\_frequency\_counter** (*channel, enabled, range, threshold\_major, threshold\_minor*)

This function is only define in the header file and it is not in the manual. This function is only valid for ps4000 and ps4000a.

**set\_no\_of\_captures** (*n\_captures*)

This function sets the number of captures to be collected in one run of rapid block mode. If you do not call this function before a run, the driver will capture only one waveform. Once a value has been set, the value remains constant unless changed.

**set\_sig\_gen\_arbitrary** (*waveform*, *sample\_frequency=None*, *offset\_voltage=0.0*,  
*pk\_to\_pk=None*, *start\_delta\_phase=None*,  
*stop\_delta\_phase=None*, *delta\_phase\_increment=0*,  
*dwll\_count=None*, *sweep\_type='up'*, *operation='off'*,  
*index\_mode='single'*, *shots=None*, *sweeps=None*,  
*trigger\_type='rising'*, *trigger\_source='None'*,  
*ext\_in\_threshold=0*)

This function programs the signal generator to produce an arbitrary waveform.

**set\_sig\_gen\_built\_in** (*offset\_voltage*, *pk\_to\_pk*, *wave\_type*, *start\_frequency*,  
*stop\_frequency*, *increment*, *dwll\_time*, *sweep\_type*, *op-*  
*eration*, *shots*, *sweeps*, *trigger\_type*, *trigger\_source*,  
*ext\_in\_threshold*)

This function sets up the signal generator to produce a signal from a list of built-in waveforms. If different start and stop frequencies are specified, the device will sweep either up, down or up and down.

**set\_sig\_gen\_built\_in\_v2** (*offset\_voltage=0.0*, *pk\_to\_pk=1.0*, *wave\_type='sine'*,  
*start\_frequency=1.0*, *stop\_frequency=None*, *in-*  
*crement=0.1*, *dwll\_time=1.0*, *sweep\_type='up'*,  
*operation='off'*, *shots=None*, *sweeps=None*,  
*trigger\_type='rising'*, *trigger\_source='None'*,  
*ext\_in\_threshold=0*)

This function is an upgraded version of `set_sig_gen_built_in()` with double-precision frequency arguments for more precise control at low frequencies.

This function is invalid for ps4000 and ps4000a.

### Parameters

- **offset\_voltage** (*float*) – The voltage offset, in **volts**, to be applied to the waveform.
- **pk\_to\_pk** (*float*) – The peak-to-peak voltage, in **volts**, of the waveform signal.
- **wave\_type** (*enum.IntEnum*) – The type of waveform to be generated. A `WaveType` enum.
- **start\_frequency** (*float*) – The frequency that the signal generator will initially produce.
- **stop\_frequency** (*float*) – The frequency at which the sweep reverses direction or returns to the initial frequency.
- **increment** (*float*) – The amount of frequency increase or decrease in sweep mode.
- **dwll\_time** (*float*) – The time, in seconds, for which the sweep stays at each frequency.
- **sweep\_type** (*enum.IntEnum*) – Whether the frequency will sweep from *start\_frequency* to *stop\_frequency*, or in the opposite direction, or repeatedly reverse direction. One of: UP, DOWN, UPDOWN, DOWNUP
- **operation** (*enum.IntEnum*) – The type of waveform to be produced, specified by one of the following enumerated types (B models only): OFF, WHITENOISE, PRBS

- **shots** (*int* of *None*) – If *None* then start and run continuously after trigger occurs.
- **sweeps** (*int* of *None*) – If *None* then start a sweep and continue after trigger occurs.
- **trigger\_type** (*enum.IntEnum*) – The type of trigger that will be applied to the signal generator. One of: *RISING*, *FALLING*, *GATE\_HIGH*, *GATE\_LOW*.
- **trigger\_source** (*enum.IntEnum* or *None*) – The source that will trigger the signal generator. If *None* then run without waiting for trigger.
- **ext\_in\_threshold** (*int*) – Used to set trigger level for external trigger.

**set\_sig\_gen\_properties\_arbitrary** (*start\_delta\_phase*, *stop\_delta\_phase*, *delta\_phase\_increment*, *dwell\_count*, *sweep\_type*, *shots*, *sweeps*, *trigger\_type*, *trigger\_source*, *ext\_in\_threshold*, *offset\_voltage=0*, *pk\_to\_pk=-1*)

This function reprograms the arbitrary waveform generator. All values can be reprogrammed while the signal generator is waiting for a trigger.

The *offset\_voltage* and *pk\_to\_pk* arguments are only used for ps6000.

This function is invalid for ps4000 and ps5000.

**set\_sig\_gen\_properties\_built\_in** (*start\_frequency*, *stop\_frequency*, *increment*, *dwell\_time*, *sweep\_type*, *shots*, *sweeps*, *trigger\_type*, *trigger\_source*, *ext\_in\_threshold*, *offset\_voltage=0*, *pk\_to\_pk=-1*)

This function reprograms the signal generator. Values can be changed while the signal generator is waiting for a trigger.

The *offset\_voltage* and *pk\_to\_pk* arguments are only used for ps6000.

This function is invalid for ps4000 and ps5000.

**set\_simple\_trigger** (*enable*, *source*, *threshold*, *direction*, *delay*, *auto\_trigger\_ms*)

This function simplifies arming the trigger. It supports only the *LEVEL* trigger types and does not allow more than one channel to have a trigger applied to it. Any previous pulse width qualifier is cancelled.

**set\_trigger\_channel\_directions** (*a='rising'*, *b='rising'*, *c='rising'*, *d='rising'*, *ext='rising'*, *aux='rising'*)

This function sets the direction of the trigger for each channel.

ps4000a overrides this method because it has a different implementation.

**set\_trigger\_delay** (*delay*)

This function sets the post-trigger delay, which causes capture to start a defined time after the trigger event.

**sig\_gen\_arbitrary\_min\_max\_values** ()

This function returns the range of possible sample values and waveform buffer sizes that can be supplied to *set\_sig\_gen\_arbitrary* () for setting up the arbitrary waveform generator (AWG).

**sig\_gen\_frequency\_to\_phase** (*frequency, index\_mode, buffer\_length*)

This function converts a frequency to a phase count for use with the arbitrary waveform generator (AWG). The value returned depends on the length of the buffer, the index mode passed and the device model. The phase count can then be sent to the driver through `set_sig_gen_arbitrary()` or `set_sig_gen_properties_arbitrary()`.

#### Parameters

- **frequency** (`float`) – The AWG sample frequency.
- **index\_mode** (`enum.IntEnum`) – An `IndexMode` enum value or member name.
- **buffer\_length** (`int`) – The size (number of samples) of the waveform.

**Returns** `int` – The phase count.

**sig\_gen\_software\_control** (*state*)

This function causes a trigger event, or starts and stops gating. It is used when the signal generator is set to `SOFT_TRIG`.

**trigger\_within\_pre\_trigger\_samples** (*state*)

This function is in the header file, but it is not in the manual.

This function is only valid for ps4000 and ps5000a.

**set\_trigger\_channel\_conditions** (*conditions, info='clear'*)

This function sets up trigger conditions on the scope's inputs. The trigger is defined by one or more `picoscope_structs.TriggerConditions` structures that are then ORED together. Each structure is itself the AND of the states of one or more of the inputs. This AND-OR logic allows you to create any possible Boolean function of the scope's inputs.

The *info* parameter is only used for ps4000a and it is a `PS4000AConditionsInfo` enum.

**set\_trigger\_channel\_properties** (*channel\_properties, timeout=0.1, aux\_output\_enable=0*)

This function is used to enable or disable triggering and set its parameters.

#### Parameters

- **channel\_properties** (`list` of `picoscope_structs`) – A list of `TriggerChannelProperties` structures describing the requested properties.
- **timeout** (`float`) – The time, in seconds, for which the scope device will wait before collecting data if no trigger event occurs. If this is set to zero, the scope device will wait indefinitely for a trigger.
- **aux\_output\_enable** (`int`) – Zero configures the AUXIO connector as a trigger input. Any other value configures it as a trigger output. Only used by ps5000.

## msl.equipment.resources.picotech.picoscope.ps2000 module

A wrapper around the PicoScope ps2000 SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_2k3k.PicoScope2k3k`

A wrapper around the PicoScope ps2000 SDK.

**Parameters record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**FIRST\_USB = 1**

**LAST\_USB = 127**

**MAX\_UNITS = 127**

**MAX\_TIMEBASE = 19**

**PS2105\_MAX\_TIMEBASE = 20**

**PS2104\_MAX\_TIMEBASE = 19**

**PS2200\_MAX\_TIMEBASE = 23**

**MAX\_OVERSAMPLE = 256**

**PS2105\_MAX\_ETS\_CYCLES = 250**

**PS2105\_MAX\_ETS\_INTERLEAVE = 50**

**PS2104\_MAX\_ETS\_CYCLES = 125**

**PS2104\_MAX\_ETS\_INTERLEAVE = 25**

**PS2203\_MAX\_ETS\_CYCLES = 250**

**PS2203\_MAX\_ETS\_INTERLEAVE = 50**

**PS2204\_MAX\_ETS\_CYCLES = 250**

**PS2204\_MAX\_ETS\_INTERLEAVE = 40**

**PS2205\_MAX\_ETS\_CYCLES = 250**

**PS2205\_MAX\_ETS\_INTERLEAVE = 40**

**MIN\_ETS\_CYCLES\_INTERLEAVE\_RATIO = 1**

**MAX\_ETS\_CYCLES\_INTERLEAVE\_RATIO = 10**

**MIN\_SIGGEN\_FREQ = 0.0**

**MAX\_SIGGEN\_FREQ = 100000.0**

**MAX\_VALUE = 32767**

**MIN\_VALUE = -32767**

**LOST\_DATA = -32768**

**last\_button\_press ()**

This function returns the last registered state of the pushbutton on the PicoScope 2104 or 2105 PC Oscilloscope and then resets the status to zero.

**set\_led (state)**

This function turns the LED on the oscilloscope on and off, and controls its colour.

**set\_light (state)**

This function controls the white light that illuminates the probe tip on a handheld oscilloscope.

**set\_sig\_gen\_arbitrary** (*offset\_voltage*, *pk\_to\_pk*, *start\_delta\_phase*,  
*stop\_delta\_phase*, *delta\_phase\_increment*, *dwell\_count*,  
*arbitrary\_waveform\_size*, *sweep\_type*, *sweeps*)

This function programs the signal generator to produce an arbitrary waveform.

**set\_sig\_gen\_built\_in** (*offset\_voltage*, *pk\_to\_pk*, *wave\_type*, *start\_frequency*,  
*stop\_frequency*, *increment*, *dwell\_time*, *sweep\_type*,  
*sweeps*)

This function sets up the signal generator to produce a signal from a list of built-in waveforms. If different start and stop frequencies are specified, the oscilloscope will sweep either up, down or up and down.

## msl.equipment.resources.picotech.picoscope.ps2000a module

A wrapper around the PicoScope ps2000a SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000A` (*record*)  
Bases: `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi`

A wrapper around the PicoScope ps2000a SDK.

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register** *Database*.

**PS2208\_MAX\_ETS\_CYCLES = 500**

**PS2208\_MAX\_INTERLEAVE = 20**

**PS2207\_MAX\_ETS\_CYCLES = 500**

**PS2207\_MAX\_INTERLEAVE = 20**

**PS2206\_MAX\_ETS\_CYCLES = 250**

**PS2206\_MAX\_INTERLEAVE = 10**

**EXT\_MAX\_VALUE = 32767**

**EXT\_MIN\_VALUE = -32767**

**MAX\_LOGIC\_LEVEL = 32767**

**MIN\_LOGIC\_LEVEL = -32767**

**MIN\_SIG\_GEN\_FREQ = 0.0**

**MAX\_SIG\_GEN\_FREQ = 20000000.0**

**MAX\_SIG\_GEN\_BUFFER\_SIZE = 8192**

**MIN\_SIG\_GEN\_BUFFER\_SIZE = 1**

**MIN\_DWELL\_COUNT = 3**

**MAX\_SWEEPS\_SHOTS = 1073741823**

**MAX\_ANALOGUE\_OFFSET\_50MV\_200MV = 0.25**

**MIN\_ANALOGUE\_OFFSET\_50MV\_200MV = -0.25**

**MAX\_ANALOGUE\_OFFSET\_500MV\_2V = 2.5**

```

MIN_ANALOGUE_OFFSET_500MV_2V = -2.5
MAX_ANALOGUE_OFFSET_5V_20V = 20.0
MIN_ANALOGUE_OFFSET_5V_20V = -20.0
SHOT_SWEEP_TRIGGER_CONTINUOUS_RUN = 4294967295
SINE_MAX_FREQUENCY = 1000000.0
SQUARE_MAX_FREQUENCY = 1000000.0
TRIANGLE_MAX_FREQUENCY = 1000000.0
SINC_MAX_FREQUENCY = 1000000.0
RAMP_MAX_FREQUENCY = 1000000.0
HALF_SINE_MAX_FREQUENCY = 1000000.0
GAUSSIAN_MAX_FREQUENCY = 1000000.0
PRBS_MAX_FREQUENCY = 1000000.0
PRBS_MIN_FREQUENCY = 0.03
MIN_FREQUENCY = 0.03

```

**set\_digital\_analog\_trigger\_operand** (*operand*)

This function is define in the header file, but it is not in the manual.

**set\_trigger\_digital\_port\_properties** (*directions*)

This function will set the individual Digital channels trigger directions. Each trigger direction consists of a channel name and a direction. If the channel is not included in the array of `PS2000ADigitalChannelDirections` the driver assumes the digital channel's trigger direction is `PS2000A_DIGITAL_DONT_CARE`.

### msl.equipment.resources.picotech.picoscope.ps3000 module

A wrapper around the PicoScope ps3000 SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_2k3k.PicoScope2k3k`

A wrapper around the PicoScope ps3000 SDK.

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register** *Database*.

```

FIRST_USB = 1
LAST_USB = 127
MAX_UNITS = 127
PS3206_MAX_TIMEBASE = 21
PS3205_MAX_TIMEBASE = 20
PS3204_MAX_TIMEBASE = 19
PS3224_MAX_TIMEBASE = 19

```

PS3223\_MAX\_TIMEBASE = 19  
PS3424\_MAX\_TIMEBASE = 19  
PS3423\_MAX\_TIMEBASE = 19  
PS3225\_MAX\_TIMEBASE = 18  
PS3226\_MAX\_TIMEBASE = 19  
PS3425\_MAX\_TIMEBASE = 19  
PS3426\_MAX\_TIMEBASE = 19  
MAX\_OVERSAMPLE = 256  
MAX\_VALUE = 32767  
MIN\_VALUE = -32767  
LOST\_DATA = -32768  
MIN\_SIGGEN\_FREQ = 0.093  
MAX\_SIGGEN\_FREQ = 1000000  
PS3206\_MAX\_ETS\_CYCLES = 500  
PS3206\_MAX\_ETS\_INTERLEAVE = 100  
PS3205\_MAX\_ETS\_CYCLES = 250  
PS3205\_MAX\_ETS\_INTERLEAVE = 50  
PS3204\_MAX\_ETS\_CYCLES = 125  
PS3204\_MAX\_ETS\_INTERLEAVE = 25  
MAX\_ETS\_CYCLES\_INTERLEAVE\_RATIO = 10  
MIN\_ETS\_CYCLES\_INTERLEAVE\_RATIO = 1  
PS300\_MAX\_ETS\_SAMPLES = 100000  
MAX\_PULSE\_WIDTH\_QUALIFIER\_COUNT = 16777215  
MAX\_HOLDOFF\_COUNT = 8388607

**release\_stream\_buffer()**

Not found in the manual, but it is in the header file.

**save\_streaming\_data** (*lp\_callback\_func, data\_buffer\_size*)

This function sends all available streaming data to the *lp\_callback\_func* callback function in your application. Your callback function decides what to do with the data.

**set\_siggen** (*wave\_type, start\_frequency, stop\_frequency, increment, dwell\_time, repeat, dual\_slope*)

This function is used to enable or disable the signal generator and sweep functions.

**streaming\_ns\_get\_interval\_stateless** (*n\_channels*)

Not found in the manual, but it is in the header file.



**msl.equipment.resources.picotech.picoscope.ps3000a module**

A wrapper around the PicoScope ps3000a SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000A` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi`

A wrapper around the PicoScope ps3000a SDK.

Parameters **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

```

MAX_OVERSAMPLE = 256
PS3207A_MAX_ETS_CYCLES = 500
PS3207A_MAX_INTERLEAVE = 40
PS3206A_MAX_ETS_CYCLES = 500
PS3206A_MAX_INTERLEAVE = 40
PS3206MSO_MAX_INTERLEAVE = 80
PS3205A_MAX_ETS_CYCLES = 250
PS3205A_MAX_INTERLEAVE = 20
PS3205MSO_MAX_INTERLEAVE = 40
PS3204A_MAX_ETS_CYCLES = 125
PS3204A_MAX_INTERLEAVE = 10
PS3204MSO_MAX_INTERLEAVE = 20
EXT_MAX_VALUE = 32767
EXT_MIN_VALUE = -32767
MAX_LOGIC_LEVEL = 32767
MIN_LOGIC_LEVEL = -32767
MIN_SIG_GEN_FREQ = 0.0
MAX_SIG_GEN_FREQ = 20000000.0
PS3207B_MAX_SIG_GEN_BUFFER_SIZE = 32768
PS3206B_MAX_SIG_GEN_BUFFER_SIZE = 16384
MAX_SIG_GEN_BUFFER_SIZE = 8192
MIN_SIG_GEN_BUFFER_SIZE = 1
MIN_DWELL_COUNT = 3
MAX_SWEEPS_SHOTS = 1073741823
MAX_ANALOGUE_OFFSET_50MV_200MV = 0.25
MIN_ANALOGUE_OFFSET_50MV_200MV = -0.25
MAX_ANALOGUE_OFFSET_500MV_2V = 2.5

```

```
MIN_ANALOGUE_OFFSET_500MV_2V = -2.5
MAX_ANALOGUE_OFFSET_5V_20V = 20.0
MIN_ANALOGUE_OFFSET_5V_20V = -20.0
SHOT_SWEEP_TRIGGER_CONTINUOUS_RUN = 4294967295
SINE_MAX_FREQUENCY = 1000000.0
SQUARE_MAX_FREQUENCY = 1000000.0
TRIANGLE_MAX_FREQUENCY = 1000000.0
SINC_MAX_FREQUENCY = 1000000.0
RAMP_MAX_FREQUENCY = 1000000.0
HALF_SINE_MAX_FREQUENCY = 1000000.0
GAUSSIAN_MAX_FREQUENCY = 1000000.0
PRBS_MAX_FREQUENCY = 1000000.0
PRBS_MIN_FREQUENCY = 0.03
MIN_FREQUENCY = 0.03
```

**get\_max\_ets\_values ()**

This function returns the maximum number of cycles and maximum interleaving factor that can be used for the selected scope device in ETS mode. These values are the upper limits for the `etsCycles` and `etsInterleave` arguments supplied to `set_ets ()`.

**get\_trigger\_info\_bulk** (*from\_segment\_index, to\_segment\_index*)

This function returns trigger information in rapid block mode.

Populates the `PS3000ATriggerInfo` structure.

**set\_pulse\_width\_digital\_port\_properties** (*directions*)

This function will set the individual digital channels' pulse-width trigger directions. Each trigger direction consists of a channel name and a direction. If the channel is not included in the array of `PS3000ADigitalChannelDirections` the driver assumes the digital channel's pulse-width trigger direction is `PS3000A_DIGITAL_DONT_CARE`.

**set\_pulse\_width\_qualifier\_v2** (*conditions, direction, lower, upper, pulse\_width\_type*)

This function sets up pulse-width qualification, which can be used on its own for pulse width triggering or combined with level triggering or window triggering to produce more complex triggers. The pulse-width qualifier is set by defining one or more structures that are then ORed together. Each structure is itself the AND of the states of one or more of the inputs. This AND-OR logic allows you to create any possible Boolean function of the scope's inputs.

**set\_trigger\_channel\_conditions\_v2** (*conditions*)

This function sets up trigger conditions on the scope's inputs. The trigger is defined by one or more `PS3000ATriggerConditionsV2` structures that are then ORed together. Each structure is itself the AND of the states of one or more of the inputs. This AND-OR logic allows you to create any possible Boolean function of the scope's inputs.

If complex triggering is not required, use `set_simple_trigger ()`.

**set\_trigger\_digital\_port\_properties** (*directions*)

This function will set the individual digital channels' trigger directions. Each trigger direction consists of a channel name and a direction. If the channel is not included in the array of PS3000ADigitalChannelDirections the driver assumes the digital channel's trigger direction is PS3000A\_DIGITAL\_DONT\_CARE.

**msl.equipment.resources.picotech.picoscope.ps4000 module**

A wrapper around the PicoScope ps4000 SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi`

A wrapper around the PicoScope ps4000 SDK.

**Parameters** **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

```

MAX_OVERSAMPLE_12BIT = 16
MAX_OVERSAMPLE_8BIT = 256
PS4XXX_MAX_ETS_CYCLES = 400
PS4XXX_MAX_INTERLEAVE = 80
PS4262_MAX_VALUE = 32767
PS4262_MIN_VALUE = -32767
MAX_VALUE = 32764
MIN_VALUE = -32764
LOST_DATA = -32768
EXT_MAX_VALUE = 32767
EXT_MIN_VALUE = -32767
MAX_PULSE_WIDTH_QUALIFIER_COUNT = 16777215
MAX_DELAY_COUNT = 8388607
MIN_SIG_GEN_FREQ = 0.0
MAX_SIG_GEN_FREQ = 100000.0
MAX_SIG_GEN_FREQ_4262 = 20000.0
MIN_SIG_GEN_BUFFER_SIZE = 1
MAX_SIG_GEN_BUFFER_SIZE = 8192
MIN_DWELL_COUNT = 10
PS4262_MAX_WAVEFORM_BUFFER_SIZE = 4096
PS4262_MIN_DWELL_COUNT = 3
MAX_SWEEPS_SHOTS = 1073741823

```

**get\_probe()**

This function is in the header file, but it is not in the manual.

**get\_trigger\_channel\_time\_offset** (*segment\_index, channel*)

This function gets the time, as two 4-byte values, at which the trigger occurred, adjusted for the time skew of the specified channel relative to the trigger source. Call it after block-mode data has been captured or when data has been retrieved from a previous block-mode capture.

**get\_trigger\_channel\_time\_offset64** (*segment\_index, channel*)

This function gets the time, as a single 8-byte value, at which the trigger occurred, adjusted for the time skew of the specified channel relative to the trigger source. Call it after block-mode data has been captured or when data has been retrieved from a previous block-mode capture.

**get\_values\_trigger\_channel\_time\_offset\_bulk** (*from\_segment\_index, to\_segment\_index, channel*)

This function retrieves the time offset, as lower and upper 32-bit values, for a group of waveforms obtained in rapid block mode, adjusted for the time skew relative to the trigger source. The array size for `timesUpper` and `timesLower` must be greater than or equal to the number of waveform time offsets requested. The segment indexes are inclusive.

**get\_values\_trigger\_channel\_time\_offset\_bulk64** (*from\_segment\_index, to\_segment\_index, channel*)

This function retrieves the time offset, as a 64-bit integer, for a group of waveforms captured in rapid block mode, adjusted for the time skew relative to the trigger source. The array size of `times` must be greater than or equal to the number of waveform time offsets requested. The segment indexes are inclusive.

**open\_unit\_async\_ex()**

This function opens a scope device selected by serial number without blocking the calling thread. You can find out when it has finished by periodically calling `open_unit_progress()` until that function returns a non-zero value.

**open\_unit\_ex()**

This function opens a scope device. The maximum number of units that can be opened is determined by the operating system, the kernel driver and the PC's hardware.

**run\_streaming\_ex** (*sample\_interval\_time\_units, max\_pre\_trigger\_samples, max\_post\_pre\_trigger\_samples, auto\_stop, down\_sample\_ratio, down\_sample\_ratio\_mode, overview\_buffer\_size*)

This function tells the oscilloscope to start collecting data in streaming mode and with a specified data reduction mode. When data has been collected from the device it is aggregated and the values returned to the application. Call `get_streaming_latest_values()` to retrieve the data.

**set\_bw\_filter** (*channel, enable*)

This function enables or disables the bandwidth-limiting filter on the specified channel.

**set\_data\_buffer\_with\_mode** (*channel, buffer\_length, mode*)

This function registers your data buffer, for non-aggregated data, with the PicoScope 4000 driver. You need to allocate the buffer before calling this function.

**set\_data\_buffers\_with\_mode** (*channel, buffer\_length, mode*)

This function registers your data buffers, for receiving aggregated data, with the PicoScope 4000 driver. You need to allocate memory for the buffers before calling this function.

**set\_ext\_trigger\_range** (*ext\_range*)  
This function sets the range of the external trigger.

**set\_probe** (*probe, range\_*)  
This function is in the header file, but it is not in the manual.

### **msl.equipment.resources.picotech.picoscope.ps4000a module**

A wrapper around the PicoScope ps4000a SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi`

A wrapper around the PicoScope ps4000a SDK.

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register** *Database*.

**MAX\_VALUE** = 32767

**MIN\_VALUE** = -32767

**LOST\_DATA** = -32768

**EXT\_MAX\_VALUE** = 32767

**EXT\_MIN\_VALUE** = -32767

**MAX\_PULSE\_WIDTH\_QUALIFIER\_COUNT** = 16777215

**MAX\_DELAY\_COUNT** = 8388607

**MAX\_SIG\_GEN\_BUFFER\_SIZE** = 16384

**MIN\_SIG\_GEN\_BUFFER\_SIZE** = 10

**MIN\_DWELL\_COUNT** = 10

**MAX\_SWEEPS\_SHOTS** = 1073741823

**AWG\_DAC\_FREQUENCY** = 8000000.0

**AWG\_PHASE\_ACCUMULATOR** = 4294967296.0

**MAX\_ANALOGUE\_OFFSET\_50MV\_200MV** = 0.25

**MIN\_ANALOGUE\_OFFSET\_50MV\_200MV** = -0.25

**MAX\_ANALOGUE\_OFFSET\_500MV\_2V** = 2.5

**MIN\_ANALOGUE\_OFFSET\_500MV\_2V** = -2.5

**MAX\_ANALOGUE\_OFFSET\_5V\_20V** = 20.0

**MIN\_ANALOGUE\_OFFSET\_5V\_20V** = -20.0

**SINE\_MAX\_FREQUENCY** = 1000000.0

**SQUARE\_MAX\_FREQUENCY** = 1000000.0

**TRIANGLE\_MAX\_FREQUENCY** = 1000000.0

**SINC\_MAX\_FREQUENCY** = 1000000.0

**RAMP\_MAX\_FREQUENCY = 1000000.0**

**HALF\_SINE\_MAX\_FREQUENCY = 1000000.0**

**GAUSSIAN\_MAX\_FREQUENCY = 1000000.0**

**MIN\_FREQUENCY = 0.03**

**apply\_resistance\_scaling** (*channel, range\_, buffer\_length*)

This function is in the header file, but it is not in the manual.

**connect\_detect** (*sensors*)

This function is in the header file, but it is not in the manual.

**device\_meta\_data** (*meta\_type, operation, format\_*)

This function is in the header file, but it is not in the manual.

**get\_common\_mode\_overflow** ()

This function is in the header file, but it is not in the manual.

**get\_string** (*string\_value*)

This function is in the header file, but it is not in the manual.

**set\_channel\_led** (*led\_states*)

This function is in the header file, but it is not in the manual.

**set\_pulse\_width\_qualifier\_conditions** (*conditions, info*)

This function sets up the conditions for pulse width qualification, which can be used on its own for pulse width triggering or combined with window triggering to produce more complex triggers. Each call to this function creates a pulse width qualifier equal to the logical AND of the elements of the `conditions` array. Calling this function multiple times creates the logical OR of multiple AND operations. This AND-OR logic allows you to create any possible Boolean function of the scope's inputs.

Other settings of the pulse width qualifier are configured by calling `set_pulse_width_qualifier_properties()`.

**set\_pulse\_width\_qualifier\_properties** (*direction, lower, upper, pulse\_width\_type*)

This function configures the general properties of the pulse width qualifier.

**set\_trigger\_channel\_directions** (*directions*)

This function sets the direction of the trigger for the specified channels.

## msl.equipment.resources.picotech.picoscope.ps5000 module

A wrapper around the PicoScope ps5000 SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps5000.PicoScope5000` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi`

A wrapper around the PicoScope ps5000 SDK.

Parameters **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**MAX\_OVERSAMPLE\_8BIT = 256**

**MAX\_VALUE = 32512**

```

MIN_VALUE = -32512
LOST_DATA = -32768
EXT_MAX_VALUE = 32767
EXT_MIN_VALUE = -32767
MAX_PULSE_WIDTH_QUALIFIER_COUNT = 16777215
MAX_DELAY_COUNT = 8388607
MAX_SIG_GEN_BUFFER_SIZE = 8192
MIN_SIG_GEN_BUFFER_SIZE = 10
MIN_DWELL_COUNT = 10
MAX_SWEEPS_SHOTS = 1073741823
SINE_MAX_FREQUENCY = 20000000.0
SQUARE_MAX_FREQUENCY = 20000000.0
TRIANGLE_MAX_FREQUENCY = 20000000.0
SINC_MAX_FREQUENCY = 20000000.0
RAMP_MAX_FREQUENCY = 20000000.0
HALF_SINE_MAX_FREQUENCY = 20000000.0
GAUSSIAN_MAX_FREQUENCY = 20000000.0
MIN_FREQUENCY = 0.03

```

### **msl.equipment.resources.picotech.picoscope.ps5000a module**

A wrapper around the PicoScope ps5000a SDK.

```

class msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000A(record)
    Bases: msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi

```

A wrapper around the PicoScope ps5000a SDK.

**Parameters** *record* (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

```

MAX_VALUE_8BIT = 32512
MIN_VALUE_8BIT = -32512
MAX_VALUE_16BIT = 32767
MIN_VALUE_16BIT = -32767
LOST_DATA = -32768
EXT_MAX_VALUE = 32767
EXT_MIN_VALUE = -32767
MAX_PULSE_WIDTH_QUALIFIER_COUNT = 16777215

```

```
MAX_DELAY_COUNT = 8388607
PS5X42A_MAX_SIG_GEN_BUFFER_SIZE = 16384
PS5X43A_MAX_SIG_GEN_BUFFER_SIZE = 32768
PS5X44A_MAX_SIG_GEN_BUFFER_SIZE = 49152
MIN_SIG_GEN_BUFFER_SIZE = 1
MIN_DWELL_COUNT = 3
MAX_SWEEPS_SHOTS = 1073741823
AWG_DAC_FREQUENCY = 200000000.0
AWG_PHASE_ACCUMULATOR = 4294967296.0
MAX_ANALOGUE_OFFSET_50MV_200MV = 0.25
MIN_ANALOGUE_OFFSET_50MV_200MV = -0.25
MAX_ANALOGUE_OFFSET_500MV_2V = 2.5
MIN_ANALOGUE_OFFSET_500MV_2V = -2.5
MAX_ANALOGUE_OFFSET_5V_20V = 20.0
MIN_ANALOGUE_OFFSET_5V_20V = -20.0
PS5244A_MAX_ETS_CYCLES = 500
PS5244A_MAX_ETS_INTERLEAVE = 40
PS5243A_MAX_ETS_CYCLES = 250
PS5243A_MAX_ETS_INTERLEAVE = 20
PS5242A_MAX_ETS_CYCLES = 125
PS5242A_MAX_ETS_INTERLEAVE = 10
SHOT_SWEEP_TRIGGER_CONTINUOUS_RUN = 4294967295
SINE_MAX_FREQUENCY = 20000000.0
SQUARE_MAX_FREQUENCY = 20000000.0
TRIANGLE_MAX_FREQUENCY = 20000000.0
SINC_MAX_FREQUENCY = 20000000.0
RAMP_MAX_FREQUENCY = 20000000.0
HALF_SINE_MAX_FREQUENCY = 20000000.0
GAUSSIAN_MAX_FREQUENCY = 20000000.0
MIN_FREQUENCY = 0.03
EXT_MAX_VOLTAGE = 5.0
get_device_resolution()
```

Returns `enum.IntEnum` – This function retrieves the resolution the specified device will run in.



**get\_trigger\_info\_bulk** (*from\_segment\_index, to\_segment\_index*)

This function is in the header file, but it is not in the manual.

Populates the PS5000ATriggerInfo structure.

**set\_device\_resolution** (*resolution*)

This function sets the new resolution. When using 12 bits or more the memory is halved. When using 15-bit resolution only 2 channels can be enabled to capture data, and when using 16-bit resolution only one channel is available. If resolution is changed, any data captured that has not been saved will be lost. If `set_channel()` is not called, `run_block()` and `run_streaming()` may fail.

### msl.equipment.resources.picotech.picoscope.ps6000 module

A wrapper around the PicoScope ps6000 SDK.

**class** `msl.equipment.resources.picotech.picoscope.ps6000.PicoScope6000` (*record*)

Bases: `msl.equipment.resources.picotech.picoscope.picoscope_api.PicoScopeApi`

A wrapper around the PicoScope ps6000 SDK.

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register** *Database*.

**MAX\_OVERSAMPLE\_8BIT = 256**

**MAX\_VALUE = 32512**

**MIN\_VALUE = -32512**

**MAX\_PULSE\_WIDTH\_QUALIFIER\_COUNT = 16777215**

**MAX\_SIG\_GEN\_BUFFER\_SIZE = 16384**

**PS640X\_C\_D\_MAX\_SIG\_GEN\_BUFFER\_SIZE = 65536**

**MIN\_SIG\_GEN\_BUFFER\_SIZE = 1**

**MIN\_DWELL\_COUNT = 3**

**MAX\_SWEEPS\_SHOTS = 1073741823**

**MAX\_WAVEFORMS\_PER\_SECOND = 1000000**

**MAX\_ANALOGUE\_OFFSET\_50MV\_200MV = 0.5**

**MIN\_ANALOGUE\_OFFSET\_50MV\_200MV = -0.5**

**MAX\_ANALOGUE\_OFFSET\_500MV\_2V = 2.5**

**MIN\_ANALOGUE\_OFFSET\_500MV\_2V = -2.5**

**MAX\_ANALOGUE\_OFFSET\_5V\_20V = 20.0**

**MIN\_ANALOGUE\_OFFSET\_5V\_20V = -20.0**

**MAX\_ETS\_CYCLES = 250**

**MAX\_INTERLEAVE = 50**

**PRBS\_MAX\_FREQUENCY = 20000000.0**

**SINE\_MAX\_FREQUENCY = 20000000.0**

**SQUARE\_MAX\_FREQUENCY = 20000000.0**

**TRIANGLE\_MAX\_FREQUENCY = 20000000.0**

**SINC\_MAX\_FREQUENCY = 20000000.0**

**RAMP\_MAX\_FREQUENCY = 20000000.0**

**HALF\_SINE\_MAX\_FREQUENCY = 20000000.0**

**GAUSSIAN\_MAX\_FREQUENCY = 20000000.0**

**MIN\_FREQUENCY = 0.03**

**get\_values\_bulk\_async** (*start\_index*, *down\_sample\_ratio*,  
*down\_sample\_ratio\_mode*, *from\_segment\_index*,  
*to\_segment\_index*)

This function is in the header file, but it is not in the manual.

**set\_data\_buffers\_bulk** (*channel*, *buffer\_length*, *waveform*,  
*down\_sample\_ratio\_mode*)

This function tells the driver where to find the buffers for aggregated data for each waveform in rapid block mode. The number of waveforms captured is determined by the `nCaptures` argument sent to `set_no_of_captures()`. Call one of the `GetValues` functions to retrieve the data after capture. If you do not need two buffers, because you are not using aggregate mode, then you can optionally use `set_data_buffer_bulk()` instead.

**set\_external\_clock** (*frequency*, *threshold*)

This function tells the scope whether or not to use an external clock signal fed into the AUX input. The external clock can be used to synchronise one or more PicoScope 6000 units to an external source.

When the external clock input is enabled, the oscilloscope relies on the clock signal for all of its timing. The driver checks that the clock is running before starting a capture, but if the clock signal stops after the initial check, the oscilloscope will not respond to any further commands until it is powered down and back up again.

Note: if the AUX input is set as an external clock input then it cannot also be used as an external trigger input.

**set\_waveform\_limiter** (*n\_waveforms\_per\_second*)

This function is in the header file, but it is not in the manual.

## **msl.equipment.resources.picotech.picoscope.structs module**

Structures defined in the Pico Technology SDK v10.6.10.24

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProp`

Bases: `_ctypes.Structure`

**channel**

Structure/Union member

**hysteresis**

Structure/Union member

**thresholdMajor**  
Structure/Union member

**thresholdMinor**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerConditions`  
Bases: `_ctypes.Structure`

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000PwqConditions`  
Bases: `_ctypes.Structure`

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**digital**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000APwqConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**digital**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000ADigitalChannelDirec`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**direction**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerChannelProp`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**thresholdLower**  
Structure/Union member

**thresholdLowerHysteresis**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**thresholdUpper**  
Structure/Union member

**thresholdUpperHysteresis**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**hysteresis**  
Structure/Union member

**thresholdMajor**  
Structure/Union member

**thresholdMinor**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerConditions`  
Bases: `_ctypes.Structure`

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000PwqConditions`  
Bases: `_ctypes.Structure`

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerConditionsV`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**digital**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000APwqConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000APwqConditionsV2`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**digital**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannelDire`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**direction**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerChannelProp`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**thresholdLower**  
Structure/Union member

**thresholdLowerHysteresis**

Structure/Union member

**thresholdMode**

Structure/Union member

**thresholdUpper**

Structure/Union member

**thresholdUpperHysteresis**

Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerInfo`

Bases: `_ctypes.Structure`

**reserved0**

Structure/Union member

**reserved1**

Structure/Union member

**segmentIndex**

Structure/Union member

**status**

Structure/Union member

**timeStampCounter**

Structure/Union member

**timeUnits**

Structure/Union member

**triggerTime**

Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerConditions`

Bases: `_ctypes.Structure`

**aux**

Structure/Union member

**channelA**

Structure/Union member

**channelB**

Structure/Union member

**channelC**

Structure/Union member

**channelD**

Structure/Union member

**external**

Structure/Union member

**pulseWidthQualifier**

Structure/Union member



**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000PwqConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerChannelProperties`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**thresholdLower**  
Structure/Union member

**thresholdLowerHysteresis**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**thresholdUpper**  
Structure/Union member

**thresholdUpperHysteresis**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000AChannelLedSetting`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**state**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000ADirection`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**direction**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000ACondition`  
Bases: `_ctypes.Structure`

**condition**  
Structure/Union member

**source**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProp`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**thresholdLower**  
Structure/Union member

**thresholdLowerHysteresis**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**thresholdUpper**  
Structure/Union member

**thresholdUpperHysteresis**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS4000AConnectDetect`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**state**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000PwqConditions`  
Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerChannelProperties`  
Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**hysteresis**  
Structure/Union member

**thresholdMajor**  
Structure/Union member

**thresholdMinor**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerInfo`  
Bases: `_ctypes.Structure`

**reserved0**  
Structure/Union member

**reserved1**  
Structure/Union member

**segmentIndex**  
Structure/Union member

**status**  
Structure/Union member

**timeUnits**  
Structure/Union member

**triggerIndex**  
Structure/Union member

**triggerTime**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerConditions`

Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**pulseWidthQualifier**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000APwqConditions`

Bases: `_ctypes.Structure`

**aux**  
Structure/Union member

**channelA**  
Structure/Union member

**channelB**  
Structure/Union member

**channelC**  
Structure/Union member

**channelD**  
Structure/Union member

**external**  
Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerChannelProp`

Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**thresholdLower**  
Structure/Union member

**thresholdLowerHysteresis**

Structure/Union member

**thresholdMode**

Structure/Union member

**thresholdUpper**

Structure/Union member

**thresholdUpperHysteresis**

Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS6000TriggerConditions`

Bases: `_ctypes.Structure`

**aux**

Structure/Union member

**channelA**

Structure/Union member

**channelB**

Structure/Union member

**channelC**

Structure/Union member

**channelD**

Structure/Union member

**external**

Structure/Union member

**pulseWidthQualifier**

Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS6000PwqConditions`

Bases: `_ctypes.Structure`

**aux**

Structure/Union member

**channelA**

Structure/Union member

**channelB**

Structure/Union member

**channelC**

Structure/Union member

**channelD**

Structure/Union member

**external**

Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS6000TriggerChannelProp`

Bases: `_ctypes.Structure`

**channel**  
Structure/Union member

**hysteresisLower**  
Structure/Union member

**hysteresisUpper**  
Structure/Union member

**thresholdLower**  
Structure/Union member

**thresholdMode**  
Structure/Union member

**thresholdUpper**  
Structure/Union member

### **msl.equipment.resources.thorlabs package**

Wrappers around APIs from Thorlabs.

#### **Subpackages**

#### **msl.equipment.resources.thorlabs.kinesis package**

Wrapper package around the Thorlabs.MotionControl.C\_API.

The Kinesis software can be downloaded from the [Thorlabs website](#)

#### **Submodules**

#### **msl.equipment.resources.thorlabs.kinesis.api\_functions module**

C Functions defined in the Thorlabs Kinesis software v1.11.0

#### **msl.equipment.resources.thorlabs.kinesis.callbacks module**

`msl.equipment.resources.thorlabs.kinesis.callbacks.MotionControlCallback`  
A callback to register for a MotionControl message queue.

Example usage:

```
from msl.examples.equipment import EXAMPLES_DIR
from msl.equipment.database import load
from msl.equipment.resources.thorlabs import MotionControlCallback

@MotionControlCallback
def msg_callback():
    print('MotionControlCallback: ', flipper.convert_message(*flipper.
↪get_next_message()))
```

```
# The "equipment-configuration.xml" configuration file contains the
↳following element:
# <equipment alias="filter_flipper" manufacturer="Thorlabs" model=
↳"MFF101/M" serial="37871232"/>

db = load(os.path.join(EXAMPLES_DIR, 'equipment-configuration.xml'))
flipper = db.equipment['filter_flipper'].connect()
flipper.register_message_callback(msg_callback)

... do stuff with the `flipper` ...
```

alias of CFunctionType

### msl.equipment.resources.thorlabs.kinesis.enums module

Enums defined in the Thorlabs Kinesis software v1.11.0

**class** `msl.equipment.resources.thorlabs.kinesis.enums.FT_Status`

Bases: `enum.IntEnum`

An enumeration.

**FT\_OK = 0**

**FT\_InvalidHandle = 1**

**FT\_DeviceNotFound = 2**

**FT\_DeviceNotOpened = 3**

**FT\_IOError = 4**

**FT\_InsufficientResources = 5**

**FT\_InvalidParameter = 6**

**FT\_DeviceNotPresent = 7**

**FT\_IncorrectDevice = 8**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_MotorTypes`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_NotMotor = 0**

**MOT\_DCMotor = 1**

**MOT\_StepperMotor = 2**

**MOT\_BrushlessMotor = 3**

**MOT\_CustomMotor = 100**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_TravelModes`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_TravelModeUndefined = 0**

**MOT\_Linear = 1**

**MOT\_Rotational = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_TravelDirection`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_TravelDirectionDisabled = 0**

**MOT\_Forwards = 1**

**MOT\_Reverse = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_DirectionSense`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_Normal = 0**

**MOT\_Backwards = 1**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_HomeLimitSwitchDirection`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_LimitSwitchDirectionUndefined = 0**

**MOT\_ReverseLimitSwitch = 1**

**MOT\_ForwardLimitSwitch = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_JogModes`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_JogModeUndefined = 0**

**MOT\_Continuous = 1**

**MOT\_SingleStep = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_StopModes`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_StopModeUndefined = 0**

**MOT\_Immediate = 1**

**MOT\_Profiled = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_ButtonModes`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_ButtonModeUndefined = 0**

**MOT\_JogMode = 1**



**MOT\_Preset = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_VelocityProfileModes`  
Bases: `enum.IntEnum`

An enumeration.

**MOT\_Trapezoidal = 0**

**MOT\_SCurve = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_LimitSwitchModes`  
Bases: `enum.IntEnum`

An enumeration.

**MOT\_LimitSwitchModeUndefined = 0**

**MOT\_LimitSwitchIgnoreSwitch = 1**

**MOT\_LimitSwitchMakeOnContact = 2**

**MOT\_LimitSwitchBreakOnContact = 3**

**MOT\_LimitSwitchMakeOnHome = 4**

**MOT\_LimitSwitchBreakOnHome = 5**

**MOT\_PMD\_Reserved = 6**

**MOT\_LimitSwitchIgnoreSwitchSwapped = 129**

**MOT\_LimitSwitchMakeOnContactSwapped = 130**

**MOT\_LimitSwitchBreakOnContactSwapped = 131**

**MOT\_LimitSwitchMakeOnHomeSwapped = 132**

**MOT\_LimitSwitchBreakOnHomeSwapped = 133**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_LimitSwitchSWModes`  
Bases: `enum.IntEnum`

An enumeration.

**MOT\_LimitSwitchSWModeUndefined = 0**

**MOT\_LimitSwitchIgnored = 1**

**MOT\_LimitSwitchStopImmediate = 2**

**MOT\_LimitSwitchStopProfiled = 3**

**MOT\_LimitSwitchIgnored\_Rotational = 129**

**MOT\_LimitSwitchStopImmediate\_Rotational = 130**

**MOT\_LimitSwitchStopProfiled\_Rotational = 131**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_LimitsSoftwareApproachPol`  
Bases: `enum.IntEnum`

An enumeration.

**DisallowIllegalMoves = 0**

**AllowPartialMoves = 1**

**AllowAllMoves = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_CurrentLoopPhases`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_PhaseA = 0**

**MOT\_PhaseB = 1**

**MOT\_PhaseAB = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.MOT_PID_LoopMode`

Bases: `enum.IntEnum`

An enumeration.

**MOT\_PIDLoopModeDisabled = 0**

**MOT\_PIDOpenLoopMode = 1**

**MOT\_PIDClosedLoopMode = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_SignalState`

Bases: `enum.IntEnum`

An enumeration.

**NT\_BadSignal = 0**

**NT\_GoodSignal = 1**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_Mode`

Bases: `enum.IntEnum`

An enumeration.

**NT\_ModeUndefined = 0**

**NT\_Piezo = 1**

**NT\_Latch = 2**

**NT\_Tracking = 3**

**NT\_HorizontalTracking = 4**

**NT\_VerticalTracking = 5**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_ControlMode`

Bases: `enum.IntEnum`

An enumeration.

**NT\_ControlModeUndefined = 0**

**NT\_OpenLoop = 1**

**NT\_ClosedLoop = 2**

**NT\_OpenLoopSmoothed = 3**

**NT\_ClosedLoopSmoothed = 4**

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSource
```

```
    Bases: enum.IntEnum
```

```
    An enumeration.
```

```
    NT_FeedbackSourceUndefined = 0
```

```
    NT_TIA = 1
```

```
    NT_BNC_1v = 2
```

```
    NT_BNC_2v = 3
```

```
    NT_BNC_5v = 4
```

```
    NT_BNC_10v = 5
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_TIARange
```

```
    Bases: enum.IntEnum
```

```
    An enumeration.
```

```
    NT_TIARange1_3nA = 3
```

```
    NT_TIARange2_10nA = 4
```

```
    NT_TIARange3_30nA = 5
```

```
    NT_TIARange4_100nA = 6
```

```
    NT_TIARange5_300nA = 7
```

```
    NT_TIARange6_1uA = 8
```

```
    NT_TIARange7_3uA = 9
```

```
    NT_TIARange8_10uA = 10
```

```
    NT_TIARange9_30uA = 11
```

```
    NT_TIARange10_100uA = 12
```

```
    NT_TIARange11_300uA = 13
```

```
    NT_TIARange12_1mA = 14
```

```
    NT_TIARange13_3mA = 15
```

```
    NT_TIARange14_10mA = 16
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_OddOrEven
```

```
    Bases: enum.IntEnum
```

```
    An enumeration.
```

```
    NT_OddAndEven = 1
```

```
    NT_Odd = 2
```

```
    NT_Even = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_UnderOrOver
```

```
    Bases: enum.IntEnum
```

```
    An enumeration.
```

```
    NT_InRange = 1
```

**NT\_UnderRange = 2**

**NT\_OverRange = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_CircleDiameterMode`

Bases: `enum.IntEnum`

An enumeration.

**NT\_ParameterCircleMode = 1**

**NT\_AbsPowerCircleMode = 2**

**NT\_LUTCircleMode = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_CircleAdjustment`

Bases: `enum.IntEnum`

An enumeration.

**NT\_LinearCircleAdjustment = 1**

**NT\_LogCircleAdjustment = 2**

**NT\_SquareCircleAdjustment = 3**

**NT\_CubeCircleAdjustment = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_TIARangeMode`

Bases: `enum.IntEnum`

An enumeration.

**NT\_TIARangeModeUndefined = 0**

**NT\_AutoRangeAtSelected = 1**

**NT\_ManualRangeAtSelected = 2**

**NT\_ManualRangeAtParameter = 3**

**NT\_AutoRangeAtParameter = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_LowPassFrequency`

Bases: `enum.IntEnum`

An enumeration.

**NT\_LowPassNone = 0**

**NT\_LowPass\_1Hz = 1**

**NT\_LowPass\_3Hz = 2**

**NT\_LowPass\_10Hz = 3**

**NT\_LowPass\_30Hz = 4**

**NT\_LowPass\_100Hz = 5**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_VoltageRange`

Bases: `enum.IntEnum`

An enumeration.

**NT\_VoltageRangeUndefined = 0**

**NT\_VoltageRange\_5v = 1**

**NT\_VoltageRange\_10v = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_OutputVoltageRoute`

Bases: `enum.IntEnum`

An enumeration.

**NT\_SMAOnly = 1**

**NT\_HubOrSMA = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_PowerInputUnits`

Bases: `enum.IntEnum`

An enumeration.

**NT\_Amps = 0**

**NT\_Watts = 1**

**NT\_Db = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.NT_SMA_Units`

Bases: `enum.IntEnum`

An enumeration.

**NT\_Voltage = 0**

**NT\_FullRange = 1**

**NT\_UserDefined = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.BNT_CurrentLimit`

Bases: `enum.IntEnum`

An enumeration.

**NT\_CurrentLimit\_100mA = 0**

**NT\_CurrentLimit\_250mA = 1**

**NT\_CurrentLimit\_500mA = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.BNT_OutputLowPassFilter`

Bases: `enum.IntEnum`

An enumeration.

**NT\_OutputFilter\_10Hz = 0**

**NT\_OutputFilter\_100Hz = 1**

**NT\_OutputFilter\_5kHz = 2**

**NT\_OutputFilter\_None = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.BNT_FeedbackSignalSelection`

Bases: `enum.IntEnum`

An enumeration.

**NT\_FeedbackSignalDC = 0**

**NT\_FeedbackSignalAC = 65535**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.BNT_BNCTriggerModes`  
Bases: `enum.IntEnum`

An enumeration.

**NT\_BNCModeTrigger = 0**

**NT\_BNCModeLVOut = 65535**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes`  
Bases: `enum.IntEnum`

An enumeration.

**PZ\_Undefined = 0**

**PZ\_OpenLoop = 1**

**PZ\_CloseLoop = 2**

**PZ\_OpenLoopSmooth = 3**

**PZ\_CloseLoopSmooth = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PZ_InputSourceFlags`  
Bases: `enum.IntEnum`

An enumeration.

**PZ\_SoftwareOnly = 0**

**PZ\_ExternalSignal = 1**

**PZ\_Potentiometer = 2**

**PZ\_All = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PZ_OutputLUTModes`  
Bases: `enum.IntEnum`

An enumeration.

**PZ\_Continuous = 1**

**PZ\_Fixed = 2**

**PZ\_OutputTrigEnable = 4**

**PZ\_InputTrigEnable = 8**

**PZ\_OutputTrigSenseHigh = 16**

**PZ\_InputTrigSenseHigh = 32**

**PZ\_OutputGated = 64**

**PZ\_OutputTrigRepeat = 128**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_DerivFilterState`  
Bases: `enum.IntEnum`

An enumeration.

**DerivFilterOn = 1**

**DerivFilterOff = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_NotchFilterState`  
Bases: `enum.IntEnum`

An enumeration.

**NotchFilterOn = 1**

**NotchFilterOff = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_NotchFilterChannel`  
Bases: `enum.IntEnum`

An enumeration.

**NotchFilter1 = 1**

**NotchFilter2 = 2**

**NotchFilterBoth = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOControlMode`  
Bases: `enum.IntEnum`

An enumeration.

**SWOnly = 0**

**ExtBNC = 1**

**Joystick = 2**

**JoystickBnc = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOOutputMode`  
Bases: `enum.IntEnum`

An enumeration.

**HV = 1**

**PosRaw = 2**

**PosCorrected = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOOutputBandwidth`  
Bases: `enum.IntEnum`

An enumeration.

**OP\_Unfiltered = 1**

**OP\_200Hz = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOFeedbackSourceDefinition`  
Bases: `enum.IntEnum`

An enumeration.

**StrainGauge = 1**

**Capacitive = 2**

**Optical = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.PPC_DisplayIntensity`

Bases: `enum.IntEnum`

An enumeration.

**Bright = 1**

**Dim = 2**

**Off = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.FF_Positions`

Bases: `enum.IntEnum`

An enumeration.

**FF\_PositionError = 0**

**Position1 = 1**

**Position2 = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.FF_IOModes`

Bases: `enum.IntEnum`

An enumeration.

**FF\_ToggleOnPositiveEdge = 1**

**FF\_SetPositionOnPositiveEdge = 2**

**FF\_OutputHighAtSetPosition = 4**

**FF\_OutputHighWhenMoving = 8**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.FF_SignalModes`

Bases: `enum.IntEnum`

An enumeration.

**FF\_InputButton = 1**

**FF\_InputLogic = 2**

**FF\_InputSwap = 4**

**FF\_OutputLevel = 16**

**FF\_OutputPulse = 32**

**FF\_OutputSwap = 64**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KMOT_JoystickDirectionSense`

Bases: `enum.IntEnum`

An enumeration.

**KMOT\_JS\_Positive = 1**

**KMOT\_JS\_Negative = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KMOT_JoyStickMode`

Bases: `enum.IntEnum`

An enumeration.



```
KMOT_JS_Velocity = 1
```

```
KMOT_JS_Jog = 2
```

```
KMOT_JS_MoveAbsolute = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KMOT_TriggerPortMode
```

```
Bases: enum.IntEnum
```

An enumeration.

```
KMOT_TrigDisabled = 0
```

```
KMOT_TrigIn_GPI = 1
```

```
KMOT_TrigIn_RelativeMove = 2
```

```
KMOT_TrigIn_AbsoluteMove = 3
```

```
KMOT_TrigIn_Home = 4
```

```
KMOT_TrigOut_GPO = 10
```

```
KMOT_TrigOut_InMotion = 11
```

```
KMOT_TrigOut_AtMaxVelocity = 12
```

```
KMOT_TrigOut_AtPositionSteps = 13
```

```
KMOT_TrigOut_Synch = 14
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KMOT_TriggerPortPolarity
```

```
Bases: enum.IntEnum
```

An enumeration.

```
KMOT_TrigPolarityHigh = 1
```

```
KMOT_TrigPolarityLow = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.LS_InputSourceFlags
```

```
Bases: enum.IntEnum
```

An enumeration.

```
LS_SoftwareOnly = 0
```

```
LS_ExternalSignal = 1
```

```
LS_Potentiometer = 4
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.LS_DisplayUnits
```

```
Bases: enum.IntEnum
```

An enumeration.

```
LS_mAmps = 1
```

```
LS_mWatts = 2
```

```
LS_mDb = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KLS_OpMode
```

```
Bases: enum.IntEnum
```

An enumeration.

**KLS\_ConstantPower = 0**

**KLS\_ConstantCurrent = 1**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KLS_TriggerMode`

Bases: `enum.IntEnum`

An enumeration.

**KLS\_Disabled = 0**

**KLS\_Input = 1**

**KLS\_ModulationTrigger = 2**

**KLS\_SetPower = 3**

**KLS\_Output = 10**

**KLS\_LaserOn = 11**

**KLS\_InterlockEnabled = 12**

**KLS\_SetPointChange = 13**

**KLS\_HighStability = 14**

**KLS\_LowStability = 15**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KLS_TrigPolarity`

Bases: `enum.IntEnum`

An enumeration.

**KLS\_TrigPol\_High = 1**

**KLS\_TrigPol\_Low = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KPZ_JoystickDirectionSense`

Bases: `enum.IntEnum`

An enumeration.

**KPZ\_JS\_Positive = 1**

**KPZ\_JS\_Negative = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KPZ_JoystickMode`

Bases: `enum.IntEnum`

An enumeration.

**KPZ\_JS\_MoveAtVoltage = 1**

**KPZ\_JS\_JogVoltage = 2**

**KPZ\_JS\_SetVoltage = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KPZ_JoystickChangeRate`

Bases: `enum.IntEnum`

An enumeration.

**KPZ\_JS\_High = 1**

**KPZ\_JS\_Medium = 2**

**KPZ\_JS\_Low = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KPZ_TriggerPortMode`  
Bases: `enum.IntEnum`

An enumeration.

**KPZ\_TrigDisabled = 0**

**KPZ\_TrigIn\_GPI = 1**

**KPZ\_TrigIn\_VoltageStepUp = 2**

**KPZ\_TrigIn\_VoltageStepDown = 3**

**KPZ\_TrigOut\_GPO = 10**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KPZ_TriggerPortPolarity`  
Bases: `enum.IntEnum`

An enumeration.

**KPZ\_TrigPolarityHigh = 1**

**KPZ\_TrigPolarityLow = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.HubAnalogueModes`  
Bases: `enum.IntEnum`

An enumeration.

**AnalogueCh1 = 1**

**AnalogueCh2 = 2**

**ExtSignalSMA = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.QD_OperatingMode`  
Bases: `enum.IntEnum`

An enumeration.

**QD\_ModeUndefined = 0**

**QD\_Monitor = 1**

**QD\_OpenLoop = 2**

**QD\_ClosedLoop = 3**

**QD\_AutoOpenClosedLoop = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.QD_LowVoltageRoute`  
Bases: `enum.IntEnum`

An enumeration.

**QD\_RouteUndefined = 0**

**QD\_SMAOnly = 1**

**QD\_HubAndSMA = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.QD_OpenLoopHoldValues`  
Bases: `enum.IntEnum`

An enumeration.

**QD\_HoldOnZero = 1**

**QD\_HoldOnLastValue = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.QD_FilterEnable`

Bases: `enum.IntEnum`

An enumeration.

**QD\_Undefined = 0**

**QD\_Enabled = 1**

**QD\_Disabled = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.QD_KPA_TrigModes`

Bases: `enum.IntEnum`

An enumeration.

**QD\_Trig\_Disabled = 0**

**QD\_TrigIn\_GPI = 1**

**QD\_TrigIn\_LoopOpenClose = 2**

**KD\_TrigOut\_GPO = 10**

**KD\_TrigOut\_Sum = 11**

**KD\_TrigOut\_Diff = 12**

**KD\_TrigOut\_SumDiff = 13**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.QD_KPA_TrigPolarities`

Bases: `enum.IntEnum`

An enumeration.

**GD\_Trig\_High = 1**

**GD\_Trig\_Low = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.SC_OperatingModes`

Bases: `enum.IntEnum`

An enumeration.

**SC\_Manual = 1**

**SC\_Single = 2**

**SC\_Auto = 3**

**SC\_Triggered = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.SC_OperatingStates`

Bases: `enum.IntEnum`

An enumeration.

**SC\_Active = 1**

**SC\_Inactive = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.SC_SolenoidStates`

Bases: `enum.IntEnum`

An enumeration.

**SC\_SolenoidOpen = 1**

**SC\_SolenoidClosed = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KSC_TriggerPortMode`

Bases: `enum.IntEnum`

An enumeration.

**KSC\_TrigDisabled = 0**

**KSC\_TrigIn\_GPI = 1**

**KSC\_TrigOut\_GPO = 10**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.KSC_TriggerPortPolarity`

Bases: `enum.IntEnum`

An enumeration.

**KSC\_TrigPolarityHigh = 1**

**KSC\_TrigPolarityLow = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TIM_Channels`

Bases: `enum.IntEnum`

An enumeration.

**Channel1 = 1**

**Channel2 = 2**

**Channel3 = 3**

**Channel4 = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TIM_JogMode`

Bases: `enum.IntEnum`

An enumeration.

**JogContinuous = 1**

**JogStep = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TIM_ButtonsMode`

Bases: `enum.IntEnum`

An enumeration.

**Jog = 1**

**Position = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TIM_Direction`

Bases: `enum.IntEnum`

An enumeration.

**Forward = 1**

**Reverse = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.LD_InputSourceFlags`

Bases: `enum.IntEnum`

An enumeration.

**LD\_SoftwareOnly = 1**

**LD\_ExternalSignal = 2**

**LD\_Potentiometer = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.LD_DisplayUnits`

Bases: `enum.IntEnum`

An enumeration.

**LD\_ILim = 1**

**LD\_ILD = 2**

**LD\_IPD = 3**

**LD\_PLD = 4**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.LD_TIA_RANGES`

Bases: `enum.IntEnum`

An enumeration.

**LD\_TIA\_10uA = 1**

**LD\_TIA\_100uA = 2**

**LD\_TIA\_1mA = 4**

**LD\_TIA\_10mA = 8**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.LD_POLARITY`

Bases: `enum.IntEnum`

An enumeration.

**LD\_CathodeGrounded = 1**

**LD\_AnodeGrounded = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TSG_Hub_Analogue_Modes`

Bases: `enum.IntEnum`

An enumeration.

**TSG\_HubChannel1 = 1**

**TSG\_HubChannel2 = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TSG_Display_Modes`

Bases: `enum.IntEnum`

An enumeration.

**TSG\_Undefined = 0**

**TSG\_Position = 1**

**TSG\_Voltage = 2**

**TSG\_Force = 3**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TC_SensorTypes`

Bases: `enum.IntEnum`

An enumeration.

**TC\_Transducer = 0**

**TC\_TH20kOhm = 1**

**TC\_TH200kOhm = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes`

Bases: `enum.IntEnum`

An enumeration.

**TC\_ActualTemperature = 0**

**TC\_TargetTemperature = 1**

**TC\_TempDifference = 2**

**TC\_Current = 3**

### **msl.equipment.resources.thorlabs.kinesis.errors module**

Device and Low Level Error Codes defined in the Thorlabs Kinesis software v1.11.0

### **msl.equipment.resources.thorlabs.kinesis.filter\_flipper module**

This module provides all the functionality required to control a Filter Flipper (MFF101).

**class** `msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper` (*record*)

Bases: `msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl`

A wrapper around `Thorlabs.MotionControl.FilterFlipper`

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**MIN\_TRANSIT\_TIME = 300**

**MAX\_TRANSIT\_TIME = 2800**

**MIN\_PULSE\_WIDTH = 10**

**MAX\_PULSE\_WIDTH = 200**

**open()**

Open the device for communication.

**Raises** `ConnectionError` – If not successful.

**close()**

Disconnect and close the device.

**check\_connection()**

Check connection.

**Returns** `bool` – Whether the USB is listed by the FTDI controller.

**identify()**

Sends a command to the device to make it identify itself.

**get\_hardware\_info()**

Gets the hardware information from the device.

**Returns** `structs.TLI_HardwareInformation` – The hardware information.

**Raises** `ConnectionError` – If not successful.

**get\_firmware\_version()**

Gets version number of the device firmware.

**Returns** `str` – The firmware version.

**get\_software\_version()**

Gets version number of the device software.

**Returns** `str` – The device software version.

**load\_settings()**

Update device with stored settings.

The settings are read from `ThorlabsDefaultSettings.xml`, which gets created when the Kinesis software is installed.

**Raises** `ConnectionError` – If not successful.

**persist\_settings()**

Persist the devices current settings.

**Raises** `ConnectionError` – If not successful.

**get\_number\_positions()**

Get number of positions available from the device.

**Returns** `int` – The number of positions.

**home()**

Home the device.

Homing the device will set the device to a known state and determine the home position.

**Raises** `ConnectionError` – If not successful.

**move\_to\_position** (*position*, *wait=False*)

Move the device to the specified position (index).

**Parameters**

- **position** (`int`) – The required position. Must be 1 or 2.
- **wait** (`bool`) – Whether to wait until the movement is complete before returning to the calling program.

**Raises** `ConnectionError` – If not successful.



**get\_position()**

Get the current position (index).

**Returns** `int` – The current position, 1 or 2.

**get\_io\_settings()**

Gets the I/O settings from filter flipper.

**Returns** `FF_IOSettings` – The Filter Flipper I/O settings.

**Raises** `ConnectionError` – If not successful.

**request\_io\_settings()**

Requests the I/O settings from the filter flipper.

**Raises** `ConnectionError` – If not successful.

**set\_io\_settings** (*transit\_time=500, oper1=<FF\_IOModes.FF\_ToggleOnPositiveEdge: 1>, sig1=<FF\_SignalModes.FF\_InputButton: 1>, pw1=200, oper2=<FF\_IOModes.FF\_ToggleOnPositiveEdge: 1>, sig2=<FF\_SignalModes.FF\_OutputLevel: 16>, pw2=200*)

Sets the settings on filter flipper.

**Parameters**

- **transit\_time** (`int`) – Time taken to get from one position to other in milliseconds.
- **oper1** (`FF_IOModes`) – I/O 1 Operating Mode.
- **sig1** (`FF_SignalModes`) – I/O 1 Signal Mode.
- **pw1** (`int`) – Digital I/O 1 pulse width in milliseconds.
- **oper2** (`FF_IOModes`) – I/O 2 Operating Mode.
- **sig2** (`FF_SignalModes`) – I/O 2 Signal Mode.
- **pw2** (`int`) – Digital I/O 2 pulse width in milliseconds.

**Raises** `ConnectionError` – If not successful.

**get\_transit\_time()**

Gets the transit time.

**Returns** `int` – The transit time in milliseconds.

**set\_transit\_time** (*transit\_time*)

Sets the transit time.

**Parameters** **transit\_time** (`int`) – The transit time in milliseconds.

**Raises** `ConnectionError` – If not successful.

**request\_status()**

Request status bits.

This needs to be called to get the device to send it's current status.

This is called automatically if Polling is enabled for the device using `start_polling()`.

**Raises** `ConnectionError` – If not successful.

**get\_status\_bits ()**

Get the current status bits.

This returns the latest status bits received from the device. To get new status bits, use *request\_status ()* or use the polling function, *start\_polling ()*

**Returns** `int` – The status bits from the device.

**start\_polling (milliseconds)**

Starts the internal polling loop.

This function continuously requests position and status messages.

**Parameters** `milliseconds (int)` – The polling rate, in milliseconds.

**Raises** `ConnectionError` – If not successful.

**polling\_duration ()**

Gets the polling loop duration.

**Returns** `int` – The time between polls in milliseconds or 0 if polling is not active.

**stop\_polling ()**

Stops the internal polling loop.

**time\_since\_last\_msg\_received ()**

Gets the time, in milliseconds, since the last message was received.

This can be used to determine whether communications with the device is still good.

**Returns** `int` – The time, in milliseconds, since the last message was received.

**enable\_last\_msg\_timer (enable, msg\_timeout=0)**

Enables the last message monitoring timer.

This can be used to determine whether communications with the device is still good.

**Parameters**

- **enable** (`bool`) – `True` to enable monitoring otherwise `False` to disable.
- **msg\_timeout** (`int`) – The last message error timeout in ms. Set to 0 to disable.

**has\_last\_msg\_timer\_overrun ()**

Queries if the time since the last message has exceeded the `lastMsgTimeout` set by *enable\_last\_msg\_timer ()*.

This can be used to determine whether communications with the device is still good.

**Returns** `bool` – `True` if last message timer has elapsed or `False` if monitoring is not enabled or if time of last message received is less than `msg_timeout`.

**request\_settings ()**

Requests that all settings are downloaded from the device.

This function requests that the device upload all its settings to the DLL.

**Raises** `ConnectionError` – If not successful.

**clear\_message\_queue ()**

Clears the device message queue.

**register\_message\_callback** (*callback*)

Registers a callback on the message queue.

**Parameters** **callback** (*callbacks.MotionControlCallback*) – A function to be called whenever messages are received.

**message\_queue\_size** ()

Gets the size of the message queue.

**Returns** *int* – The number of messages in the queue.

**get\_next\_message** ()

Get the next Message Queue item. See *messages*.

**Returns**

- *int* – The message type.
- *int* – The message ID.
- *int* – The message data.

**Raises** *ConnectionError* – If not successful.

**wait\_for\_message** ()

Wait for next Message Queue item. See *messages*.

**Returns**

- *int* – The message type.
- *int* – The message ID.
- *int* – The message data.

**Raises** *ConnectionError* – If not successful.

### **msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors** module

This module provides all the functionality required to control a number of **Integrated Stepper Motors** including:

- Long Travel Stages (LTS150 and LTS300)
- Lab Jack (MLJ050)
- Cage Rotator (K10CR1)

**class** `msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.UnitType`

Bases: `enum.IntEnum`

An Integrated Stepper Motor device unit.

Used to `get_real_value_from_device_unit` and to `get_device_unit_from_real_value`.

**DISTANCE = 0**

**VELOCITY = 1**

**ACCELERATION = 2**

**class** `msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors`

Bases: `msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl`

A wrapper around `Thorlabs.MotionControl.IntegratedStepperMotors`

The `record.connection.properties` dictionary for a `IntegratedStepperMotors` device supports the following key-value pairs:

```
'load_settings': bool, # optional, default is True (load the default_
↳ settings when connection is created)
```

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**can\_home** ()

Can the device perform a *home* ()?

**Returns** `bool` – Whether the device can be homed.

**can\_move\_without\_homing\_first** ()

Does the device need to be *home*'d before a move can be performed?

**Returns** `bool` – Whether the device needs to be homed.

**check\_connection** ()

Check connection.

**Returns** `bool` – Whether the USB is listed by the FTDI controller.

**clear\_message\_queue** ()

Clears the device message queue.

**close** ()

Disconnect and close the device.

**disable\_channel** ()

Disable the channel so that motor can be moved by hand.

When disabled, power is removed from the motor and it can be freely moved.

**Raises** `ConnectionError` – If not successful.

**enable\_channel** ()

Enable channel for computer control.

When enabled, power is applied to the motor so it is fixed in position.

**Raises** `ConnectionError` – If not successful.

**enable\_last\_msg\_timer** (*enable*, *last\_msg\_timeout*)

Enables the last message monitoring timer.

This can be used to determine whether communications with the device is still good.

**Parameters**

- **enable** (`bool`) – `True` to enable monitoring otherwise `False` to disable.
- **last\_msg\_timeout** (`int`) – The last message error timeout in ms. Set to 0 to disable.

**get\_backlash()**

Get the backlash distance setting (used to control hysteresis).

See *get\_real\_value\_from\_device\_unit* for converting from a `device_unit` to a `real_value`.

**Returns** `int` – The backlash distance in `DeviceUnits` (see manual).

**get\_bow\_index()**

Gets the stepper motor bow index.

**Returns** `int` – The bow index.

**get\_button\_params()**

Gets the LTS button parameters.

See *get\_real\_value\_from\_device\_unit* for converting from a `device_unit` to a `real_value`.

**Returns**

- `enums.MOT_ButtonModes` – The button mode.
- `int` – The Preset position in `DeviceUnits` for the left button (when in preset mode).
- `int` – The Preset position in `DeviceUnits` for the right button (when in preset mode).
- `int` – The time that buttons need to be pressed in order to go home or to record a preset buttons defined position.

**Raises** `ConnectionError` – If not successful.

**get\_button\_params\_block()**

Get the button parameters.

**Returns** `structs.MOT_ButtonParameters` – The button parameters.

**Raises** `ConnectionError` – If not successful.

**get\_calibration\_file()**

Get calibration file for this motor.

**Returns** `str` – The filename of the calibration file.

**Raises** `ConnectionError` – If not successful.

**get\_device\_unit\_from\_real\_value(real\_value, unit\_type)**

Converts a real-world value to a device value.

Either *load\_settings()* or *set\_motor\_params\_ext()* must be called before calling this function, otherwise the returned value will always be 0.

**Parameters**

- **real\_value** (`float`) – The real-world value.
- **unit\_type** (`UnitType`) – The unit of the real-world value.

**Returns** `int` – The device value.

**Raises** `ConnectionError` – If not successful.

**get\_firmware\_version()**

Gets version number of the device firmware.

**Returns** `str` – The firmware version.

**get\_hardware\_info()**

Gets the hardware information from the device.

**Returns** `structs.TLI_HardwareInformation` – The hardware information.

**Raises** `ConnectionError` – If not successful.

**get\_hardware\_info\_block()**

Gets the hardware information in a block.

**Returns** `structs.TLI_HardwareInformation` – The hardware information.

**Raises** `ConnectionError` – If not successful.

**get\_homing\_params\_block()**

Get the homing parameters.

**Returns** `structs.MOT_HomingParameters` – The homing parameters.

**Raises** `ConnectionError` – If not successful.

**get\_homing\_velocity()**

Gets the homing velocity.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

**Returns** `int` – The homing velocity in `DeviceUnits` (see manual).

**get\_jog\_mode()**

Gets the jog mode.

**Returns**

- `enums.MOT_JogModes` – The jog mode.
- `enums.MOT_StopModes` – The stop mode.

**Raises** `ConnectionError` – If not successful.

**get\_jog\_params\_block()**

Get the jog parameters.

**Returns** `structs.MOT_JogParameters` – The jog parameters.

**Raises** `ConnectionError` – If not successful.

**get\_jog\_step\_size()**

Gets the distance to move when jogging.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

**Returns** `int` – The step size in `DeviceUnits` (see manual).

**get\_jog\_vel\_params ()**

Gets the jog velocity parameters.

See *get\_real\_value\_from\_device\_unit* for converting from a *device\_unit* to a *real\_value*.

**Returns**

- *int* – The maximum velocity in *DeviceUnits* (see manual).
- *int* – The acceleration in *DeviceUnits* (see manual).

**Raises** *ConnectionError* – If not successful.

**get\_led\_switches ()**

Get the LED indicator bits on the device.

**Returns** *int* – Sum of: 8 to indicate moving 2 to indicate end of track and 1 to flash on identify command.

**get\_limit\_switch\_params ()**

Gets the limit switch parameters.

See *get\_real\_value\_from\_device\_unit* for converting from a *device\_unit* to a *real\_value*.

**Returns**

- *enums.MOT\_LimitSwitchModes* – The clockwise hardware limit mode.
- *enums.MOT\_LimitSwitchModes* – The anticlockwise hardware limit mode.
- *int* – The position of the clockwise software limit in *DeviceUnits* (see manual).
- *int* – The position of the anticlockwise software limit in *DeviceUnits* (see manual).
- *enums.MOT\_LimitSwitchSWModes* – The soft limit mode.

**Raises** *ConnectionError* – If not successful.

**get\_limit\_switch\_params\_block ()**

Get the limit switch parameters.

**Returns** *structs.MOT\_LimitSwitchParameters* – The limit switch parameters.

**Raises** *ConnectionError* – If not successful.

**get\_motor\_params ()**

Gets the motor stage parameters.

Deprecated: calls *get\_motor\_params\_ext ()*

These parameters, when combined define the stage motion in terms of *RealWorldUnits* [millimeters or degrees]. The real-world unit is defined from  $steps\_per\_rev * gear\_box\_ratio / pitch$ .

**Returns**

- `float` – The steps per revolution.
- `float` – The gear box ratio.
- `float` – The pitch.

**Raises** `ConnectionError` – If not successful.

**get\_motor\_params\_ext()**

Gets the motor stage parameters.

These parameters, when combined define the stage motion in terms of `RealWorldUnits` [millimeters or degrees]. The real-world unit is defined from `steps_per_rev * gear_box_ratio / pitch`.

**Returns**

- `float` – The steps per revolution.
- `float` – The gear box ratio.
- `float` – The pitch.

**Raises** `ConnectionError` – If not successful.

**get\_motor\_travel\_limits()**

Gets the motor stage min and max position.

**Returns**

- `float` – The minimum position in `RealWorldUnits` [millimeters or degrees].
- `float` – The maximum position in `RealWorldUnits` [millimeters or degrees].

**Raises** `ConnectionError` – If not successful.

**get\_motor\_travel\_mode()**

Get the motor travel mode.

**Returns** `enums.MOT_TravelModes` – The travel mode.

**get\_motor\_velocity\_limits()**

Gets the motor stage maximum velocity and acceleration.

**Returns**

- `float` – The maximum velocity in `RealWorldUnits` [millimeters or degrees].
- `float` – The maximum acceleration in `RealWorldUnits` [millimeters or degrees].

**Raises** `ConnectionError` – If not successful.

**get\_move\_absolute\_position()**

Gets the move absolute position.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

**Returns** `int` – The absolute position in `DeviceUnits` (see manual).



**get\_move\_relative\_distance()**

Gets the move relative distance.

See *get\_real\_value\_from\_device\_unit* for converting from a `device_unit` to a `real_value`.

**Returns** `int` – The move relative position in `DeviceUnits` (see manual).

**get\_next\_message()**

Get the next Message Queue item. See *messages*.

**Returns**

- `int` – The message type.
- `int` – The message ID.
- `int` – The message data.

**Raises** `ConnectionError` – If not successful.

**get\_number\_positions()**

Get number of positions.

This function will get the maximum position reachable by the device. The motor may need to be set to its *home()* position before this parameter can be used.

**Returns** `int` – The number of positions.

**get\_position()**

Get the current position.

See *get\_real\_value\_from\_device\_unit* for converting from a `device_unit` to a `real_value`.

**Returns** `index(int)` – The position in `DeviceUnits` (see manual).

**get\_position\_counter()**

Get the position counter.

The position counter is identical to the position parameter. The position counter is set to zero when homing is complete.

See *get\_real\_value\_from\_device\_unit* for converting from a `device_unit` to a `real_value`.

**Returns** `int` – The position counter in `DeviceUnits` (see manual).

**get\_potentiometer\_params(index)**

Gets the potentiometer parameters for the LTS.

See *get\_real\_value\_from\_device\_unit* for converting from a `device_unit` to a `real_value`.

**Parameters** `index(int)` – The potentiometer index to be read.

**Returns**

- `int` – The potentiometer threshold, range 0 to 127.
- `int` – The velocity in `DeviceUnits` for the current potentiometer threshold.

**Raises** `ConnectionError` – If not successful.

**get\_potentiometer\_params\_block()**

Get the potentiometer parameters.

**Returns** *structs.MOT\_PotentiometerSteps* – The potentiometer parameters.

**Raises** *ConnectionError* – If not successful.

**get\_power\_params()**

Gets the power parameters for the stepper motor.

**Returns** *structs.MOT\_PowerParameters* – The power parameters.

**Raises** *ConnectionError* – If not successful.

**get\_real\_value\_from\_device\_unit(device\_value, unit\_type)**

Converts a device value to a real-world value.

Either *load\_settings()* or *set\_motor\_params\_ext()* must be called before calling this function, otherwise the returned value will always be 0.0.

**Parameters**

- **device\_value** (*int*) – The device value.
- **unit\_type** (*UnitType*) – The unit of the device value.

**Returns** *float* – The real-world value.

**Raises** *ConnectionError* – If not successful.

**get\_soft\_limit\_mode()**

Gets the software limits mode.

**Returns** *enums.MOT\_LimitsSoftwareApproachPolicy* – The software limits mode.

**get\_software\_version()**

Gets version number of the device software.

**Returns** *str* – The device software version.

**get\_stage\_axis\_max\_pos()**

Gets the LTS Motor maximum stage position.

See *get\_real\_value\_from\_device\_unit* for converting from a *device\_unit* to a *real\_value*.

**Returns** *int* – The maximum position in *DeviceUnits* (see manual).

**get\_stage\_axis\_min\_pos()**

Gets the LTS Motor minimum stage position.

See *get\_real\_value\_from\_device\_unit* for converting from a *device\_unit* to a *real\_value*.

**Returns** *int* – The minimum position in *DeviceUnits* (see manual).

**get\_status\_bits()**

Get the current status bits.

This returns the latest status bits received from the device. To get new status bits, use `request_status_bits()` or use `request_status()` or use the polling functions, `start_polling()`.

**Returns** `int` – The status bits from the device.

**get\_trigger\_switches()**

Gets the trigger switch bits.

**Returns** `int` – 8 bits indicating action on trigger input and events to trigger electronic output.

**get\_vel\_params()**

Gets the move velocity parameters.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

**Returns**

- **max\_velocity** (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**get\_vel\_params\_block()**

Get the move velocity parameters.

**Returns** `structs.MOT_VelocityParameters` – The velocity parameters.

**Raises** `ConnectionError` – If not successful.

**has\_last\_msg\_timer\_overrun()**

Queries if the time since the last message has exceeded the `lastMsgTimeout` set by `enable_last_msg_timer()`.

This can be used to determine whether communications with the device is still good.

**Returns** `bool` – `True` if last message timer has elapsed or `False` if monitoring is not enabled or if time of last message received is less than `lastMsgTimeout`.

**home** (`wait=True`)

Home the device.

Homing the device will set the device to a known state and determine the home position.

**Parameters** `wait` (`bool`) – Wait until the device has been homed before returning to the calling program.

**Raises** `ConnectionError` – If not successful.

**identify()**

Sends a command to the device to make it identify itself.

**is\_calibration\_active()**

Is a calibration file active for this motor?

**Returns** `bool` – Whether a calibration file is active.

**load\_settings ()**

Update device with stored settings.

The settings are read from `ThorlabsDefaultSettings.xml`, which gets created when the Kinesis software is installed.

**Raises** `ConnectionError` – If not successful.

**message\_queue\_size ()**

Gets the size of the message queue.

**Returns** `int` – The number of messages in the queue.

**move\_absolute ()**

Moves the device to the position defined in `set_move_absolute_position ()`.

**Raises** `ConnectionError` – If not successful.

**move\_at\_velocity (direction)**

Start moving at the current velocity in the specified direction.

**Parameters** `direction` (`enums.MOT_TravelDirection`) – The required direction of travel as a `enums.MOT_TravelDirection` enum value or member name.

**Raises** `ConnectionError` – If not successful.

**move\_jog (jog\_direction)**

Perform a jog.

**Parameters** `jog_direction` (`enums.MOT_TravelDirection`) – The jog direction as a `enums.MOT_TravelDirection` enum value or member name.

**Raises** `ConnectionError` – If not successful.

**move\_relative (displacement)**

Move the motor by a relative amount.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters** `displacement` (`int`) – Signed displacement in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**move\_relative\_distance ()**

Moves the device by a relative distance defined by `set_move_relative_distance ()`.

**Raises** `ConnectionError` – If not successful.

**move\_to\_position (index, wait=True)**

Move the device to the specified position (`index`).

The motor may need to be set to its `home ()` position before a position can be set.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters**

- **index** (*int*) – The position in `DeviceUnits` (see manual).
- **wait** (*bool*) –

**Raises** `ConnectionError` – If not successful.

**needs\_homing** ()

Does the device need to be *home*'d before a move can be performed?

Deprecated: calls `can_move_without_homing_first ()` instead.

**Returns** `bool` – Whether the device needs to be homed.

**open** ()

Open the device for communication.

**Raises** `ConnectionError` – If not successful.

**persist\_settings** ()

Persist the devices current settings.

**Raises** `ConnectionError` – If not successful.

**polling\_duration** ()

Gets the polling loop duration.

**Returns** `int` – The time between polls in milliseconds or 0 if polling is not active.

**register\_message\_callback** (*callback*)

Registers a callback on the message queue.

**Parameters** **callback** (`callbacks.MotionControlCallback`) – A function to be called whenever messages are received.

**request\_backlash** ()

Requests the backlash.

**Raises** `ConnectionError` – If not successful.

**request\_bow\_index** ()

Requests the stepper motor bow index.

**Raises** `ConnectionError` – If not successful.

**request\_button\_params** ()

Requests the LTS button parameters.

**Raises** `ConnectionError` – If not successful.

**request\_homing\_params** ()

Requests the homing parameters.

**Raises** `ConnectionError` – If not successful.

**request\_jog\_params** ()

Requests the jog parameters.

**Raises** `ConnectionError` – If not successful.

**request\_limit\_switch\_params** ()

Requests the limit switch parameters.

**Raises** `ConnectionError` – If not successful.

**request\_move\_absolute\_position()**

Requests the position of next absolute move.

**Raises** `ConnectionError` – If not successful.

**request\_move\_relative\_distance()**

Requests the relative move distance.

**Raises** `ConnectionError` – If not successful.

**request\_position()**

Requests the current position.

This needs to be called to get the device to send it's current position. Note, this is called automatically if `Polling` is enabled for the device using `start_polling()`.

**Raises** `ConnectionError` – If not successful.

**request\_potentiometer\_params()**

Requests the potentiometer parameters.

**Raises** `ConnectionError` – If not successful.

**request\_power\_params()**

Requests the power parameters.

**Raises** `ConnectionError` – If not successful.

**request\_settings()**

Requests that all settings are downloaded from the device.

This function requests that the device upload all it's settings to the DLL.

**Raises** `ConnectionError` – If not successful.

**request\_status()**

Request position and status bits.

This needs to be called to get the device to send it's current status. Note, this is called automatically if `Polling` is enabled for the device using `start_polling()`.

**Raises** `ConnectionError` – If not successful.

**request\_status\_bits()**

Request the status bits which identify the current motor state.

This needs to be called to get the device to send it's current status bits. Note, this is called automatically if `Polling` is enabled for the device using `start_polling()`.

**Raises** `ConnectionError` – If not successful.

**request\_trigger\_switches()**

Requests the trigger switch bits.

**Raises** `ConnectionError` – If not successful.

**request\_vel\_params()**

Requests the velocity parameters.

**Raises** `ConnectionError` – If not successful.

**reset\_stage\_to\_defaults()**

Reset the stage settings to defaults.

**Raises** `ConnectionError` – If not successful.

**set\_backlash** (*distance*)

Sets the backlash distance (used to control hysteresis).

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters** **distance** (`int`) – The backlash distance in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_bow\_index** (*bow\_index*)

Sets the stepper motor bow index.

**Parameters** **bow\_index** (`int`) – The bow index.

**Raises** `ConnectionError` – If not successful.

**set\_button\_params** (*button\_mode, left\_button\_position, right\_button\_position*)

Sets the LTS button parameters.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

#### Parameters

- **button\_mode** (`enums.MOT_ButtonModes`) – The button mode as a `enums.MOT_ButtonModes` enum value or member name.
- **left\_button\_position** (`int`) – The Preset position in `DeviceUnits` for the left button (when in preset mode).
- **right\_button\_position** (`int`) – The Preset position in `DeviceUnits` for the right button (when in preset mode).

**Raises** `ConnectionError` – If not successful.

**set\_button\_params\_block** (*mode, left\_button, right\_button, timeout*)

Set the button parameters.

#### Parameters

- **mode** (`enums.MOT_ButtonModes`) – The mode of operation of the device buttons as a `enums.MOT_ButtonModes` enum value or member name.
- **left\_button** (`int`) – Position in encoder counts to go to when left button is pressed.
- **right\_button** (`int`) – Position in encoder counts to go to when right button is pressed.
- **timeout** (`int`) – The Time a button needs to be held down for to record the position as a preset.

**Raises** `ConnectionError` – If not successful.

**set\_calibration\_file** (*path, enabled*)

Set the calibration file for this motor.

#### Parameters

- **path** (*str*) – The path to a calibration file to load.
- **enabled** (*bool*) – *True* to enable, *False* to disable.

**Raises** `FileNotFoundError` – If the *path* does not exist.

**set\_direction** (*reverse*)

Sets the motor direction sense.

This function is used because some actuators use have directions of motion reversed. This parameter will tell the system to reverse the direction sense when moving, jogging etc.

**Parameters** **reverse** (*bool*) – If *True* then directions will be swapped on these moves.

**set\_homing\_params\_block** (*direction, limit, velocity, offset*)

Set the homing parameters.

#### Parameters

- **direction** (*enums.MOT\_TravelDirection*) – The Homing direction sense as a *enums.MOT\_TravelDirection* enum value or member name.
- **limit** (*enums.MOT\_HomeLimitSwitchDirection*) – The limit switch direction as a *enums.MOT\_HomeLimitSwitchDirection* enum value or member name.
- **velocity** (*int*) – The velocity in small indivisible units.
- **offset** (*int*) – Distance of home from limit in small indivisible units.

**Raises** `ConnectionError` – If not successful.

**set\_homing\_velocity** (*velocity*)

Sets the homing velocity.

See `get_device_unit_from_real_value` for converting from a *real\_value* to a *device\_unit*.

**Parameters** **velocity** (*int*) – The homing velocity in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_jog\_mode** (*mode, stop\_mode*)

Sets the jog mode.

#### Parameters

- **mode** (*enums.MOT\_JogModes*) – The jog mode, as a *enums.MOT\_JogModes* enum value or member name.
- **stop\_mode** (*enums.MOT\_StopModes*) – The stop mode, as a *enums.MOT\_StopModes* enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_jog\_params\_block** (*jog\_params*)

Set the jog parameters.

**Parameters** **jog\_params** (*structs.MOT\_JogParameters*) – The jog parameters.



**Raises**

- `ConnectionError` – If not successful.
- `TypeError` – If the data type of `jog_params` is not `structs.MOT_JogParameters`

**set\_jog\_step\_size** (*step\_size*)

Sets the distance to move on jogging.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters** `step_size` (`int`) – The step size in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_jog\_vel\_params** (*max\_velocity, acceleration*)

Sets jog velocity parameters.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters**

- `max_velocity` (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- `acceleration` (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_led\_switches** (*led\_switches*)

Set the LED indicator bits on the device.

**Parameters** `led_switches` (`int`) – Sum of: 8 to indicate moving 2 to indicate end of track and 1 to flash on identify command.

**Raises** `ConnectionError` – If not successful.

**set\_limit\_switch\_params** (*cw\_lim, ccw\_lim, cw\_pos, ccw\_pos, soft\_limit\_mode*)

Sets the limit switch parameters.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters**

- `cw_lim` (`enums.MOT_LimitSwitchModes`) – The clockwise hardware limit mode as a `enums.MOT_LimitSwitchModes` enum value or member name.
- `ccw_lim` (`enums.MOT_LimitSwitchModes`) – The anticlockwise hardware limit mode as a `enums.MOT_LimitSwitchModes` enum value or member name.
- `cw_pos` (`int`) – The position of the clockwise software limit in `DeviceUnits` (see manual).
- `ccw_pos` (`int`) – The position of the anticlockwise software limit in `DeviceUnits` (see manual).

- **soft\_limit\_mode** (*enums.MOT\_LimitSwitchSWModes*) – The soft limit mode as a *enums.MOT\_LimitSwitchSWModes* enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_limit\_switch\_params\_block** (*cw\_limit, ccw\_limit, cw\_pos, ccw\_pos, mode*)

Set the limit switch parameters.

See *get\_device\_unit\_from\_real\_value* for converting from a *real\_value* to a *device\_unit*.

#### Parameters

- **cw\_limit** (*enums.MOT\_LimitSwitchModes*) – The clockwise hardware limit as a *enums.MOT\_LimitSwitchModes* enum value or member name.
- **ccw\_limit** (*int*) – The anticlockwise hardware limit as a *enums.MOT\_LimitSwitchModes* enum value or member name.
- **cw\_pos** (*int*) – The position of clockwise software limit in *DeviceUnits*.
- **ccw\_pos** (*int*) – The position of anticlockwise software limit in *DeviceUnits*.
- **mode** (*enums.MOT\_LimitSwitchSWModes*) – Actions to take when software limit is detected as a *enums.MOT\_LimitSwitchSWModes* enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_limits\_software\_approach\_policy** (*policy*)

Sets the software limits mode.

**Parameters** *policy* (*enums.MOT\_LimitsSoftwareApproachPolicy*)

– The soft limit mode as a *enums.MOT\_LimitsSoftwareApproachPolicy* enum value or member name.

**set\_motor\_params** (*steps\_per\_rev, gear\_box\_ratio, pitch*)

Sets the motor stage parameters.

Deprecated: calls *set\_motor\_params\_ext* ()

These parameters, when combined, define the stage motion in terms of *RealWorldUnits* [millimeters or degrees]. The real-world unit is defined from  $steps\_per\_rev * gear\_box\_ratio / pitch$ .

See *get\_real\_value\_from\_device\_unit* for converting from a *device\_unit* to a *real\_value*.

#### Parameters

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises** `ConnectionError` – If not successful.

**set\_motor\_params\_ext** (*steps\_per\_rev, gear\_box\_ratio, pitch*)

Sets the motor stage parameters.

These parameters, when combined, define the stage motion in terms of `RealWorldUnits` [millimeters or degrees]. The real-world unit is defined from `steps_per_rev * gear_box_ratio / pitch`.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

#### Parameters

- **steps\_per\_rev** (`float`) – The steps per revolution.
- **gear\_box\_ratio** (`float`) – The gear box ratio.
- **pitch** (`float`) – The pitch.

**Raises** `ConnectionError` – If not successful.

**set\_motor\_travel\_limits** (*min\_position, max\_position*)

Sets the motor stage min and max position.

These define the range of travel for the stage.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

#### Parameters

- **min\_position** (`float`) – The minimum position in `RealWorldUnits` [millimeters or degrees].
- **max\_position** (`float`) – The maximum position in `RealWorldUnits` [millimeters or degrees].

**Raises** `ConnectionError` – If not successful.

**set\_motor\_travel\_mode** (*travel\_mode*)

Set the motor travel mode.

**Parameters** **travel\_mode** (`enums.MOT_TravelModes`) – The travel mode as a `enums.MOT_TravelModes` enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_motor\_velocity\_limits** (*max\_velocity, max\_acceleration*)

Sets the motor stage maximum velocity and acceleration.

See `get_real_value_from_device_unit` for converting from a `device_unit` to a `real_value`.

#### Parameters

- **max\_velocity** (`float`) – The maximum velocity in `RealWorldUnits` [millimeters or degrees].
- **max\_acceleration** (`float`) – The maximum acceleration in `RealWorldUnits` [millimeters or degrees].

**Raises** `ConnectionError` – If not successful.

**set\_move\_absolute\_position** (*position*)

Sets the move absolute position.

See *get\_device\_unit\_from\_real\_value* for converting from a *real\_value* to a *device\_unit*.

**Parameters** *position* (*int*) – The absolute position in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_move\_relative\_distance** (*distance*)

Sets the move relative distance.

See *get\_device\_unit\_from\_real\_value* for converting from a *real\_value* to a *device\_unit*.

**Parameters** *distance* (*int*) – The relative position in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_position\_counter** (*count*)

Set the position counter.

Setting the position counter will locate the current position. Setting the position counter will effectively define the home position of a motor.

See *get\_device\_unit\_from\_real\_value* for converting from a *real\_value* to a *device\_unit*.

**Parameters** *count* (*int*) – The position counter in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_potentiometer\_params** (*index*, *threshold*, *velocity*)

Sets the potentiometer parameters for the LTS.

See *get\_device\_unit\_from\_real\_value* for converting from a *real\_value* to a *device\_unit*.

**Parameters**

- **index** (*int*) – The potentiometer index to be stored.
- **threshold** (*int*) – The potentiometer threshold, range 0 to 127.
- **velocity** (*int*) – The velocity in `DeviceUnits` for the current potentiometer threshold.

**Raises**

- `ConnectionError` – If not successful.
- `ValueError` – If the value of *threshold* is out of range.

**set\_potentiometer\_params\_block** (*params*)

Set the potentiometer parameters.

**Parameters** *params* (*structs.MOT\_PotentiometerSteps*) – The potentiometer parameters.

**Raises** `ConnectionError` – If not successful.

**set\_power\_params** (*rest, move*)

Sets the power parameters for the stepper motor.

**Parameters**

- **rest** (`int`) – Percentage of full power to give while not moving (0 - 100).
- **move** (`int`) – Percentage of full power to give while moving (0 - 100).

**Raises** `ConnectionError` – If not successful.

**set\_stage\_axis\_limits** (*min\_position, max\_position*)

Sets the stage axis position limits.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters**

- **min\_position** (`int`) – The minimum position in `DeviceUnits` (see manual).
- **max\_position** (`int`) – The maximum position in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_trigger\_switches** (*indicator\_bits*)

Sets the trigger switch bits.

**Parameters** **indicator\_bits** (`int`) – Sets the 8 bits indicating action on trigger input and events to trigger electronic output.

**Raises** `ConnectionError` – If not successful.

**set\_vel\_params** (*max\_velocity, acceleration*)

Sets the move velocity parameters.

See `get_device_unit_from_real_value` for converting from a `real_value` to a `device_unit`.

**Parameters**

- **max\_velocity** (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises** `ConnectionError` – If not successful.

**set\_vel\_params\_block** (*min\_velocity, max\_velocity, acceleration*)

Set the move velocity parameters.

**Parameters**

- **min\_velocity** (`int`) – The minimum velocity.
- **max\_velocity** (`int`) – The maximum velocity.
- **acceleration** (`int`) – The acceleration.

**Raises** `ConnectionError` – If not successful.

**start\_polling** (*milliseconds*)  
Starts the internal polling loop.

This function continuously requests position and status messages.

**Parameters** `milliseconds` (*int*) – The polling rate, in milliseconds.

**Raises** `ConnectionError` – If not successful.

**stop\_immediate** ()  
Stop the current move immediately (with risk of losing track of position).

**Raises** `ConnectionError` – If not successful.

**stop\_polling** ()  
Stops the internal polling loop.

**stop\_profiled** ()  
Stop the current move using the current velocity profile.

**Raises** `ConnectionError` – If not successful.

**time\_since\_last\_msg\_received** ()  
Gets the time, in milliseconds, since the last message was received.

This can be used to determine whether communications with the device is still good.

**Returns** *int* – The time, in milliseconds, since the last message was received.

**wait\_for\_message** ()  
Wait for next Message Queue item. See *messages*.

**Returns**

- *int* – The message type.
- *int* – The message ID.
- *int* – The message data.

**Raises** `ConnectionError` – If not successful.

## **msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid module**

This module provides all the functionality required to control a KCube Solenoid (KSC101).

**class** `msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KCubeSolenoid` (*record*)  
Bases: `msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl`

A wrapper around `Thorlabs.MotionControl.KCube.Solenoid`

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register** *Database*.

**check\_connection** ()  
Check connection.

**Returns** *bool* – Whether the USB is listed by the FTDI controller.

**clear\_message\_queue()**  
Clears the device message queue.

**close()**  
Disconnect and close the device.

**enable\_last\_msg\_timer(enable, msg\_timeout=0)**  
Enables the last message monitoring timer.

This can be used to determine whether communications with the device is still good.

#### Parameters

- **enable** (*bool*) – `True` to enable monitoring otherwise `False` to disable.
- **msg\_timeout** (*int*) – The last message error timeout in ms. Set to 0 to disable.

**get\_cycle\_params()**  
Gets the cycle parameters.

#### Returns

- *int* – The *On Time* parameter. Range 250 to 100,000,000 in steps of 1 milliseconds (0.250s to 10,000s).
- *int* – The *Off Time* parameter. Range 250 to 100,000,000 in steps of 1 milliseconds (0.250s to 10,000s).
- *int* – The *Number of Cycles* parameter. Range 0 to 1000,000 where 0 represents unlimited.

**Raises** `ConnectionError` – If not successful.

**get\_cycle\_params\_block()**  
Get the cycle parameters.

**Returns** `structs.SC_CycleParameters` – The cycle parameters.

**Raises** `ConnectionError` – If not successful.

**get\_digital\_outputs()**  
Gets the digital output bits.

**Returns** *int* – Bit mask of states of the 4 digital output pins.

**get\_hardware\_info()**  
Gets the hardware information from the device.

**Returns** `structs.TLI_HardwareInformation` – The hardware information.

**Raises** `ConnectionError` – If not successful.

**get\_hardware\_info\_block()**  
Gets the hardware information in a block.

**Returns** `structs.TLI_HardwareInformation` – The hardware information.

**Raises** `ConnectionError` – If not successful.

**get\_hub\_bay()**

Gets the hub bay number this device is fitted to.

**Returns** `int` – Either the number, or 0x00 if unknown, or 0xff if not on a hub.

**get\_led\_switches()**

Get the LED indicator bits on cube.

**Returns** `int` – Sum of: 8 to indicate moving 2 to indicate end of track and 1 to flash on identify command.

**get\_mmi\_params()**

Get the MMI Parameters for the KCube Display Interface.

Deprecated: calls `get_mmi_params_ext()`

**Returns**

- `int` – The display intensity, range 0 to 100%.
- `int` – The display timeout, range 0 to 480 in minutes (0 is off, otherwise the inactivity period before dimming the display).
- `int` – The display dimmed intensity, range 0 to 10 (after the timeout period the device display will dim).

**Raises** `ConnectionError` – If not successful.

**get\_mmi\_params\_ext()**

Get the MMI Parameters for the KCube Display Interface.

**Returns**

- `int` – The display intensity, range 0 to 100%.
- `int` – The display timeout, range 0 to 480 in minutes (0 is off, otherwise the inactivity period before dimming the display).
- `int` – The display dimmed intensity, range 0 to 10 (after the timeout period the device display will dim).

**Raises** `ConnectionError` – If not successful.

**get\_next\_message()**

Get the next Message Queue item. See *messages*.

**Returns**

- `int` – The message type.
- `int` – The message ID.
- `int` – The message data.

**Raises** `ConnectionError` – If not successful.

**get\_operating\_mode()**

Gets the Operating Mode.

**Returns** `enums.SC_OperatingModes` – The current operating mode.

**get\_operating\_state()**

Gets the current operating state.



**Returns** *enums.SC\_OperatingStates* – The current operating state.

**get\_software\_version()**

Gets version number of the device software.

**Returns** *str* – The device software version.

**get\_solenoid\_state()**

Gets the current solenoid state.

**Returns** *enums.SC\_SolenoidStates* – The current solenoid state.

**get\_status\_bits()**

Get the current status bits.

This returns the latest status bits received from the device. To get new status bits, use *request\_status()* or use the polling function, *start\_polling()*

**Returns** *int* – The status bits from the device.

**get\_trigger\_config\_params()**

Get the Trigger Configuration Parameters.

**Returns**

- *enums.KSC\_TriggerPortMode* – The trigger 1 mode.
- *enums.KSC\_TriggerPortPolarity* – The trigger 1 polarity.
- *enums.KSC\_TriggerPortMode* – The trigger 2 mode.
- *enums.KSC\_TriggerPortPolarity* – The trigger 2 polarity.

**Raises** *ConnectionError* – If not successful.

**get\_trigger\_config\_params\_block()**

Gets the trigger configuration parameters block.

**Returns** *structs.KSC\_TriggerConfig* – Options for controlling the trigger configuration.

**Raises** *ConnectionError* – If not successful.

**has\_last\_msg\_timer\_overrun()**

Queries if the time since the last message has exceeded the *lastMsgTimeout* set by *enable\_last\_msg\_timer()*.

This can be used to determine whether communications with the device is still good.

**Returns** *bool* – *True* if last message timer has elapsed or *False* if monitoring is not enabled or if time of last message received is less than *lastMsgTimeout*.

**identify()**

Sends a command to the device to make it identify itself.

**load\_settings()**

Update device with stored settings.

The settings are read from *ThorlabsDefaultSettings.xml*, which gets created when the Kinesis software is installed.

**Raises** *ConnectionError* – If not successful.

**message\_queue\_size ()**

Gets the size of the message queue.

**Returns** `int` – The number of messages in the queue.

**open ()**

Open the device for communication.

**Raises** `ConnectionError` – If not successful.

**persist\_settings ()**

Persist the devices current settings.

**Raises** `ConnectionError` – If not successful.

**polling\_duration ()**

Gets the polling loop duration.

**Returns** `int` – The time between polls in milliseconds or 0 if polling is not active.

**register\_message\_callback (callback)**

Registers a callback on the message queue.

**Parameters** `callback` (`callbacks.MotionControlCallback`) – A function to be called whenever messages are received.

**request\_cycle\_params ()**

Requests the cycle parameters.

**Raises** `ConnectionError` – If not successful.

**request\_digital\_outputs ()**

Requests the digital output bits.

**Raises** `ConnectionError` – If not successful.

**request\_hub\_bay ()**

Requests the hub bay number this device is fitted to.

**Raises** `ConnectionError` – If not successful.

**request\_led\_switches ()**

Requests the LED indicator bits on the cube.

**Raises** `ConnectionError` – If not successful.

**request\_mmi\_params ()**

Requests the MMI Parameters for the KCube Display Interface.

**Raises** `ConnectionError` – If not successful.

**request\_operating\_mode ()**

Requests the operating mode.

**Raises** `ConnectionError` – If not successful.

**request\_operating\_state ()**

Requests the operating state.

**Raises** `ConnectionError` – If not successful.

**request\_settings ()**

Requests that all settings are download from device.

This function requests that the device upload all it's settings to the DLL.

**Raises** `ConnectionError` – If not successful.

#### **request\_status()**

Requests the status from the device.

This needs to be called to get the device to send it's current status bits. Note, this is called automatically if `Polling` is enabled for the device using `start_polling()`.

**Raises** `ConnectionError` – If not successful.

#### **request\_status\_bits()**

Request the status bits which identify the current motor state.

This needs to be called to get the device to send it's current status bits. Note, this is called automatically if `Polling` is enabled for the device using `start_polling()`.

**Raises** `ConnectionError` – If not successful.

#### **request\_trigger\_config\_params()**

Requests the Trigger Configuration Parameters.

**Raises** `ConnectionError` – If not successful.

#### **set\_cycle\_params(*on\_time, off\_time, num\_cycles*)**

Sets the cycle parameters.

##### **Parameters**

- **on\_time** (*int*) – The On Time parameter. Range 250 to 100,000,000 in steps of 1 milliseconds (0.250s to 10,000s).
- **off\_time** (*int*) – The Off Time parameter. Range 250 to 100,000,000 in steps of 1 milliseconds (0.250s to 10,000s).
- **num\_cycles** (*int*) – The Number of Cycles parameter Range 0 to 1,000,000 where 0 represent unlimited.

##### **Raises**

- `ValueError` – If any of the input parameters are out of range.
- `ConnectionError` – If not successful.

#### **set\_cycle\_params\_block(*cycle\_params*)**

Sets the cycle parameters.

**Parameters** **cycle\_params** (*structs.SC\_CycleParameters*) – The new cycle parameters.

**Raises** `ConnectionError` – If not successful.

#### **set\_digital\_outputs(*outputs\_bits*)**

Sets the digital output bits.

**Parameters** **outputs\_bits** (*int*) – Bit mask to set the states of the 4 digital output pins.

**Raises** `ConnectionError` – If not successful.

#### **set\_led\_switches(*led\_switches*)**

Set the LED indicator bits on the cube.

**Parameters** `led_switches` (`int`) – Sum of: 8 to indicate moving 2 to indicate end of track and 1 to flash on identify command.

**Raises** `ConnectionError` – If not successful.

**set\_mmi\_params** (`display_intensity`)

Set the MMI Parameters for the KCube Display Interface.

Deprecated: superceded by `set_mmi_params_ext()`

**Parameters** `display_intensity` (`int`) – The display intensity, range 0 to 100%.

**Raises**

- `ValueError` – If the value of `display_intensity` is out of range.
- `ConnectionError` – If not successful.

**set\_mmi\_params\_ext** (`intensity`, `timeout`, `dim_intensity`)

Set the MMI Parameters for the KCube Display Interface.

**Parameters**

- **intensity** (`int`) – The display intensity, range 0 to 100%.
- **timeout** (`int`) – The display timeout, range 0 to 480 in minutes (0 is off, otherwise the inactivity period before dimming the display).
- **dim\_intensity** (`int`) – The display dimmed intensity, range 0 to 10 (after the timeout period the device display will dim).

**Raises**

- `ValueError` – If any of the input parameters are out of range.
- `ConnectionError` – If not successful.

**set\_operating\_mode** (`mode`)

Sets the operating mode.

**Parameters** `mode` (`enums.SC_OperatingModes`) – The required operating mode as a `SC_OperatingModes` enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_operating\_state** (`state`)

Sets the operating state.

**Parameters** `state` (`enums.SC_OperatingStates`) – The required operating state as a `SC_OperatingStates` enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_trigger\_config\_params** (`mode1`, `polarity1`, `mode2`, `polarity2`)

**Parameters**

- **mode1** (`enums.KSC_TriggerPortMode`) – The trigger 1 mode as a `KSC_TriggerPortMode` enum value or member name.
- **polarity1** (`enums.KSC_TriggerPortPolarity`) – The trigger 1 polarity as a `KSC_TriggerPortPolarity` enum value or member name.

- **mode2** (*enums.KSC\_TriggerPortMode*) – The trigger 2 mode as a *KSC\_TriggerPortMode* enum value or member name.
- **polarity2** (*enums.KSC\_TriggerPortPolarity*) – The trigger 2 polarity as a *KSC\_TriggerPortPolarity* enum value or member name.

**Raises** `ConnectionError` – If not successful.

**set\_trigger\_config\_params\_block** (*trigger\_config\_params*)

Sets the trigger configuration parameters block.

**Parameters** **trigger\_config\_params** (*structs.KSC\_TriggerConfig*) – Options for controlling the trigger configuration.

**Raises** `ConnectionError` – If not successful.

**start\_polling** (*milliseconds*)

Starts the internal polling loop.

This function continuously requests position and status messages.

**Parameters** **milliseconds** (*int*) – The polling rate, in milliseconds.

**Raises** `ConnectionError` – If not successful.

**stop\_polling** ()

Stops the internal polling loop.

**time\_since\_last\_msg\_received** ()

Gets the time, in milliseconds, since the last message was received.

This can be used to determine whether communications with the device is still good.

**Returns** *int* – The time, in milliseconds, since the last message was received.

**wait\_for\_message** ()

Wait for next Message Queue item. See *messages*.

**Returns**

- *int* – The message type.
- *int* – The message ID.
- *int* – The message data.

**Raises** `ConnectionError` – If not successful.

### **msl.equipment.resources.thorlabs.kinesis.messages module**

Device Message Queue defined in `Thorlabs.MotionControl.C_API`.

The device message queue allows the internal events raised by the device to be monitored by the DLLs owner.

The device raises many different events, usually associated with a change of state.

These messages are temporarily stored in the DLL and can be accessed using the appropriate message functions.

**msl.equipment.resources.thorlabs.kinesis.motion\_control module**

Base **Thorlabs.MotionControl** class.

`msl.equipment.resources.thorlabs.kinesis.motion_control.device_manager()`  
Returns a reference to the DeviceManager library.

The `Thorlabs.MotionControl.DeviceManager.dll` library must be available on `os.environ['PATH']`.

**Returns** `ctypes.CDLL` – A reference to the library.

**class** `msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl` (*record*, *api\_function*)

Bases: `msl.equipment.connection_msl.ConnectionSDK`

Base **Thorlabs.MotionControl** class.

**Parameters**

- **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.
- **api\_function** (*api\_functions*) – An API function list from *api\_functions* that the subclass is a wrapper around.

**Raises** `ConnectionError` – If a connection to the device cannot be established.

**Benchtop\_Brushless\_Motor = 73**  
Benchtop Brushless Motor device ID

**Benchtop\_NanoTrak = 22**  
Benchtop NanoTrak device ID

**Benchtop\_Piezo\_1\_Channel = 41**  
Benchtop Piezo 1-Channel device ID

**Benchtop\_Piezo\_3\_Channel = 71**  
Benchtop Piezo 3-Channel device ID

**Benchtop\_Stepper\_Motor\_1\_Channel = 40**  
Benchtop Stepper Motor 1-Channel device ID

**Benchtop\_Stepper\_Motor\_3\_Channel = 70**  
Benchtop Stepper Motor 3-Channel device ID

**Filter\_Flipper = 37**  
Filter Flipper device ID

**Filter\_Wheel = 47**  
Filter Wheel device ID

**KCube\_Brushless\_Motor = 28**  
KCube Brushless Motor device ID

**KCube\_DC\_Servo = 27**  
KCube DC Servo device ID

**KCube\_LaserSource = 56**  
KCube Laser Source device ID

**KCube\_Piezo = 29**  
KCube Piezo device ID

**KCube\_Solenoid = 68**  
KCube Solenoid device ID

**KCube\_Stepper\_Motor = 26**  
KCube Stepper Motor device ID

**Long\_Travel\_Stage = 45**  
Long Travel Stage device ID

**Cage\_Rotator = 55**  
Cage Rotator device ID

**LabJack\_490 = 46**  
LabJack 490 device ID

**LabJack\_050 = 49**  
LabJack 050 device ID

**Modular\_NanoTrak = 52**  
Modular NanoTrak device ID

**Modular\_Piezo = 51**  
Modular Piezo device ID

**Modular\_Stepper\_Motor = 50**  
Modular Stepper Motor device ID

**TCube\_Brushless\_Motor = 67**  
TCube Brushless Motor device ID

**TCube\_DC\_Servo = 83**  
TCube DC Servo device ID

**TCube\_Inertial\_Motor = 65**  
TCube Inertial Motor device ID

**TCube\_LaserSource = 86**  
TCube Laser Source device ID

**TCube\_LaserDiode = 64**  
TCube Laser Diode device ID

**TCube\_NanoTrak = 82**  
TCube NanoTrak device ID

**TCube\_Quad = 89**  
TCube Quad device ID

**TCube\_Solenoid = 85**  
TCube Solenoid device ID

**TCube\_Stepper\_Motor = 80**  
TCube Stepper\_Motor device ID

**TCube\_Strain\_Gauge = 84**  
TCube Strain Gauge device ID

**TCube\_TEC = 87**

TCube TEC device ID

**SERIAL\_NUMBER\_BUFFER\_SIZE = 500**

**errcheck\_api** (*result, func, args*)

The API function returns OK if the function call was successful.

**errcheck\_true** (*result, func, args*)

The API function returns `True` if the function call was successful.

**disconnect** ()

Disconnect and close the device.

**static build\_device\_list** ()

Build the device list.

This function builds an internal collection of all devices found on a USB port that are not currently open.

---

**Note:** If a device is open, it will not appear in the list until the device has been closed.

---

**Raises** `ConnectionError` – If the device list cannot be built.

**static get\_device\_list\_size** ()

`int`: The number of devices in the device list.

**static get\_device\_list** (*\*device\_ids*)

Get the contents of the device list which match the supplied device IDs.

**Parameters** `device_ids` (`int`) – A sequence of device ID's.

**Returns** `list` of `str` – A list of device serial numbers for the specified device ID(s).

**Raises** `ConnectionError` – If there was an error getting the device list.

**static get\_device\_info** (*serial\_number*)

Get the device information from a USB port.

The device info is read from the USB port not from the device itself.

**Parameters** `serial_number` (`str`) – The serial number of the device.

**Returns** `structs.TLI_DeviceInfo` – A DeviceInfo structure.

**Raises** `ConnectionError` – If there was an error getting the device information.

**static to\_version** (*dword*)

Convert the firmware or software number to a string.

The number is made up of 4-byte parts.

See the `get_firmware_version()` or the `get_software_version()` method of the appropriate Thorlabs MotionControl subclass.

**Parameters** `dword` (`int`) – The firmware or software number.

**Returns** `str` – The string representation of the version number.



**static convert\_message** (*msg\_type*, *msg\_id*, *msg\_data*)

Converts the message into a string representation.

See the *get\_next\_message()* or the *wait\_for\_message()* method of the appropriate Thorlabs MotionControl subclass.

#### Parameters

- **msg\_type** (*int*) – The message type defines the device type which raised the message.
- **msg\_id** (*int*) – The message ID for the *msg\_type*.
- **msg\_data** (*int*) – The message data.

**Returns** *str* – The message as a string, with the type, id and data separated by a semicolon.

### msl.equipment.resources.thorlabs.kinesis.structs module

Structs defined in the Thorlabs Kinesis software v1.11.0

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TLI_DeviceInfo`

Bases: `_ctypes.Structure`

**PID**

Structure/Union member

**description**

Structure/Union member

**isCustomType**

Structure/Union member

**isKnownType**

Structure/Union member

**isLaser**

Structure/Union member

**isPiezoDevice**

Structure/Union member

**isRack**

Structure/Union member

**maxChannels**

Structure/Union member

**motorType**

Structure/Union member

**serialNo**

Structure/Union member

**typeID**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TLI_HardwareInformation`

Bases: `_ctypes.Structure`

**deviceDependantData**  
Structure/Union member

**firmwareVersion**  
Structure/Union member

**hardwareVersion**  
Structure/Union member

**modelName**  
Structure/Union member

**modificationState**  
Structure/Union member

**notes**  
Structure/Union member

**numChannels**  
Structure/Union member

**serialNumber**  
Structure/Union member

**type**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_VelocityParameters`  
Bases: `_ctypes.Structure`

**acceleration**  
Structure/Union member

**maxVelocity**  
Structure/Union member

**minVelocity**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_JogParameters`  
Bases: `_ctypes.Structure`

**mode**  
Structure/Union member

**stepSize**  
Structure/Union member

**stopMode**  
Structure/Union member

**velParams**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_HomingParameters`  
Bases: `_ctypes.Structure`

**direction**  
Structure/Union member

**limitSwitch**  
Structure/Union member

**offsetDistance**  
Structure/Union member

**velocity**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_VelocityProfileParameters`  
Bases: `_ctypes.Structure`

**jerk**  
Structure/Union member

**lastNotUsed**  
Structure/Union member

**mode**  
Structure/Union member

**notUsed**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_StageAxisParameters`  
Bases: `_ctypes.Structure`

**axisID**  
Structure/Union member

**countsPerUnit**  
Structure/Union member

**maxAcceleration**  
Structure/Union member

**maxDeceleration**  
Structure/Union member

**maxPosition**  
Structure/Union member

**maxVelocity**  
Structure/Union member

**minPosition**  
Structure/Union member

**partNumber**  
Structure/Union member

**reserved1**  
Structure/Union member

**reserved2**  
Structure/Union member

**reserved3**  
Structure/Union member

**reserved4**  
Structure/Union member

**reserved5**  
Structure/Union member

**reserved6**  
Structure/Union member

**reserved7**  
Structure/Union member

**reserved8**  
Structure/Union member

**serialNumber**  
Structure/Union member

**stageID**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_JoystickParameters`  
Bases: `_ctypes.Structure`

**directionSense**  
Structure/Union member

**highGearAcceleration**  
Structure/Union member

**highGearMaxVelocity**  
Structure/Union member

**lowGearAcceleration**  
Structure/Union member

**lowGearMaxVelocity**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_BrushlessPositionLoopPa`  
Bases: `_ctypes.Structure`

**accelerationFeedForward**  
Structure/Union member

**derivativeRecalculationTime**  
Structure/Union member

**differentialGain**  
Structure/Union member

**factorForOutput**  
Structure/Union member

**integralGain**  
Structure/Union member

**integralLimit**  
Structure/Union member

**lastNotUsed**  
Structure/Union member

**notUsed**  
Structure/Union member

**positionErrorLimit**  
Structure/Union member

**proportionalGain**  
Structure/Union member

**velocityFeedForward**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_BrushlessTrackSettlePar`  
Bases: `_ctypes.Structure`

**lastNotUsed**  
Structure/Union member

**maxTrackingError**  
Structure/Union member

**notUsed**  
Structure/Union member

**settledError**  
Structure/Union member

**time**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_BrushlessCurrentLoopPar`  
Bases: `_ctypes.Structure`

**deadErrorBand**  
Structure/Union member

**feedForward**  
Structure/Union member

**integralGain**  
Structure/Union member

**integralLimit**  
Structure/Union member

**lastNotUsed**  
Structure/Union member

**notUsed**  
Structure/Union member

**phase**  
Structure/Union member

**proportionalGain**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_BrushlessElectricOutput`

Bases: `_ctypes.Structure`

**continuousCurrentLimit**

Structure/Union member

**excessEnergyLimit**

Structure/Union member

**lastNotUsed**

Structure/Union member

**motorSignalBias**

Structure/Union member

**motorSignalLimit**

Structure/Union member

**notUsed**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_LimitSwitchParameters`

Bases: `_ctypes.Structure`

**anticlockwiseHardwareLimit**

Structure/Union member

**anticlockwisePosition**

Structure/Union member

**clockwiseHardwareLimit**

Structure/Union member

**clockwisePosition**

Structure/Union member

**softLimitMode**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_PowerParameters`

Bases: `_ctypes.Structure`

**movePercentage**

Structure/Union member

**restPercentage**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_DC_PIDParameters`

Bases: `_ctypes.Structure`

**differentialGain**

Structure/Union member

**integralGain**

Structure/Union member

**integralLimit**

Structure/Union member

**parameterFilter**  
Structure/Union member

**proportionalGain**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.BNT_IO_Settings`  
Bases: `_ctypes.Structure`

**BNCTriggerOrLowVoltageOut**  
Structure/Union member

**amplifierCurrentLimit**  
Structure/Union member

**amplifierLowPassFilter**  
Structure/Union member

**channel**  
Structure/Union member

**feedbackSignal**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_HVComponent`  
Bases: `_ctypes.Structure`

**horizontalComponent**  
Structure/Union member

**verticalComponent**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_CircleParameters`  
Bases: `_ctypes.Structure`

**algorithmAdjustment**  
Structure/Union member

**diameter**  
Structure/Union member

**maxDiameter**  
Structure/Union member

**minDiameter**  
Structure/Union member

**mode**  
Structure/Union member

**samplesPerRevolution**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_CircleDiameterLUT`  
Bases: `_ctypes.Structure`

**LUTdiameter**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_TIARangeParameters`  
Bases: `_ctypes.Structure`

**changeToOddOrEven**  
Structure/Union member

**downLimit**  
Structure/Union member

**mode**  
Structure/Union member

**newRange**  
Structure/Union member

**settleSamples**  
Structure/Union member

**upLimit**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_LowPassFilterParameters`  
Bases: `_ctypes.Structure`

**param1**  
Structure/Union member

**param2**  
Structure/Union member

**param3**  
Structure/Union member

**param4**  
Structure/Union member

**param5**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_TIAReading`  
Bases: `_ctypes.Structure`

**absoluteReading**  
Structure/Union member

**relativeReading**  
Structure/Union member

**selectedRange**  
Structure/Union member

**underOrOverRead**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_IOSettings`  
Bases: `_ctypes.Structure`

**lowVoltageOutOfRange**  
Structure/Union member



**lowVoltageOutputRoute**

Structure/Union member

**notYetInUse**

Structure/Union member

**unused**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.NT_GainParameters`

Bases: `_ctypes.Structure`

**controlMode**

Structure/Union member

**gain**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.PZ_FeedbackLoopConstants`

Bases: `_ctypes.Structure`

**integralTerm**

Structure/Union member

**proportionalTerm**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.PZ_LUTWaveParameters`

Bases: `_ctypes.Structure`

**LUTValueDelay**

Structure/Union member

**cycleLength**

Structure/Union member

**mode**

Structure/Union member

**numCycles**

Structure/Union member

**numOutTriggerRepeat**

Structure/Union member

**outTriggerDuration**

Structure/Union member

**outTriggerStart**

Structure/Union member

**postCycleDelay**

Structure/Union member

**preCycleDelay**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.PPC_PIDConsts`

Bases: `_ctypes.Structure`

**PIDConstsD**  
Structure/Union member

**PIDConstsDFc**  
Structure/Union member

**PIDConstsI**  
Structure/Union member

**PIDConstsP**  
Structure/Union member

**PIDDerivFilterOn**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.PPC_NotchParams`  
Bases: `_ctypes.Structure`

**filter1Fc**  
Structure/Union member

**filter1Q**  
Structure/Union member

**filter2Fc**  
Structure/Union member

**filter2Q**  
Structure/Union member

**filterNo**  
Structure/Union member

**notchFilter1On**  
Structure/Union member

**notchFilter2On**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.PPC_IOSettings`  
Bases: `_ctypes.Structure`

**FPBrightness**  
Structure/Union member

**controlSrc**  
Structure/Union member

**feedbackSrc**  
Structure/Union member

**monitorOPBandwidth**  
Structure/Union member

**monitorOPSig**  
Structure/Union member

**reserved1**  
Structure/Union member

---

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_PIDLoopEncoderParams`

Bases: `_ctypes.Structure`

**PIDOutputLimit**

Structure/Union member

**PIDTolerance**

Structure/Union member

**differentialGain**

Structure/Union member

**integralGain**

Structure/Union member

**loopMode**

Structure/Union member

**proportionalGain**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.FF_IOSettings`

Bases: `_ctypes.Structure`

**ADCspeedValue**

Structure/Union member

**digIO1OperMode**

Structure/Union member

**digIO1PulseWidth**

Structure/Union member

**digIO1SignalMode**

Structure/Union member

**digIO2OperMode**

Structure/Union member

**digIO2PulseWidth**

Structure/Union member

**digIO2SignalMode**

Structure/Union member

**reserved1**

Structure/Union member

**reserved2**

Structure/Union member

**transitTime**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_ButtonParameters`

Bases: `_ctypes.Structure`

**buttonMode**

Structure/Union member

**leftButtonPosition**  
Structure/Union member

**rightButtonPosition**  
Structure/Union member

**timeout**  
Structure/Union member

**unused**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_PotentiometerStep`  
Bases: `_ctypes.Structure`

**thresholdDeflection**  
Structure/Union member

**velocity**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.MOT_PotentiometerSteps`  
Bases: `_ctypes.Structure`

**potentiometerStepParameters**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KMOT_MMIParams`  
Bases: `_ctypes.Structure`

**DisplayDimIntensity**  
Structure/Union member

**DisplayIntensity**  
Structure/Union member

**DisplayTimeout**  
Structure/Union member

**JoystickAcceleration**  
Structure/Union member

**JoystickDirectionSense**  
Structure/Union member

**JoystickMaxVelocity**  
Structure/Union member

**JoystickMode**  
Structure/Union member

**PresetPos1**  
Structure/Union member

**PresetPos2**  
Structure/Union member

**reserved**  
Structure/Union member

---

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerConfig`  
Bases: `_ctypes.Structure`

**Trigger1Mode**  
Structure/Union member

**Trigger1Polarity**  
Structure/Union member

**Trigger2Mode**  
Structure/Union member

**Trigger2Polarity**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams`  
Bases: `_ctypes.Structure`

**CycleCount**  
Structure/Union member

**TriggerIntervalFwd**  
Structure/Union member

**TriggerIntervalRev**  
Structure/Union member

**TriggerPulseCountFwd**  
Structure/Union member

**TriggerPulseCountRev**  
Structure/Union member

**TriggerPulseWidth**  
Structure/Union member

**TriggerStartPositionFwd**  
Structure/Union member

**TriggerStartPositionRev**  
Structure/Union member

**reserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KLS_MMIPParams`  
Bases: `_ctypes.Structure`

**displayIntensity**  
Structure/Union member

**reserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KLS_TrigIOPParams`  
Bases: `_ctypes.Structure`

**model**  
Structure/Union member

**mode2**  
Structure/Union member

**polarity1**  
Structure/Union member

**polarity2**  
Structure/Union member

**power1**  
Structure/Union member

**power2**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TPZ_IOSettings`  
Bases: `_ctypes.Structure`

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KPZ_MMIParams`  
Bases: `_ctypes.Structure`

**DisplayDimIntensity**  
Structure/Union member

**DisplayIntensity**  
Structure/Union member

**DisplayTimeout**  
Structure/Union member

**JoystickDirectionSense**  
Structure/Union member

**JoystickMode**  
Structure/Union member

**PresetPos1**  
Structure/Union member

**PresetPos2**  
Structure/Union member

**VoltageAdjustRate**  
Structure/Union member

**VoltageStep**  
Structure/Union member

**reserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KPZ_TriggerConfig`  
Bases: `_ctypes.Structure`

**Trigger1Mode**  
Structure/Union member

**Trigger1Polarity**  
Structure/Union member

**Trigger2Mode**  
Structure/Union member

**Trigger2Polarity**  
Structure/Union member

**reserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_LoopParameters`  
Bases: `_ctypes.Structure`

**differentialGain**  
Structure/Union member

**integralGain**  
Structure/Union member

**lowPassFilterCutOffFreq**  
Structure/Union member

**lowPassFilterEnabled**  
Structure/Union member

**notchFilterCenterFrequency**  
Structure/Union member

**notchFilterEnabled**  
Structure/Union member

**notchFilterQ**  
Structure/Union member

**proportionalGain**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_PIDParameters`  
Bases: `_ctypes.Structure`

**differentialGain**  
Structure/Union member

**integralGain**  
Structure/Union member

**proportionalGain**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_LowPassFilterParameters`  
Bases: `_ctypes.Structure`

**lowPassFilterCutOffFreq**  
Structure/Union member

**lowPassFilterEnabled**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_NotchFilterParameters`  
Bases: `_ctypes.Structure`

**notchFilterCenterFrequency**

Structure/Union member

**notchFilterEnabled**

Structure/Union member

**notchFilterQ**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_PositionDemandParameters`

Bases: `_ctypes.Structure`

**lowVoltageOutputRoute**

Structure/Union member

**maxXdemand**

Structure/Union member

**maxYdemand**

Structure/Union member

**minXdemand**

Structure/Union member

**minYdemand**

Structure/Union member

**openLoopOption**

Structure/Union member

**xFeedbackSignedGain**

Structure/Union member

**yFeedbackSignedGain**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_Position`

Bases: `_ctypes.Structure`

**x**

Structure/Union member

**y**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_Readings`

Bases: `_ctypes.Structure`

**demandedPos**

Structure/Union member

**posDifference**

Structure/Union member

**sum**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_TrigIOConfig`

Bases: `_ctypes.Structure`



**trig1DiffThreshold**  
Structure/Union member

**trig1Mode**  
Structure/Union member

**trig1Polarity**  
Structure/Union member

**trig1SumMax**  
Structure/Union member

**trig1SumMin**  
Structure/Union member

**trig2DiffThreshold**  
Structure/Union member

**trig2Mode**  
Structure/Union member

**trig2Polarity**  
Structure/Union member

**trig2SumMax**  
Structure/Union member

**trig2SumMin**  
Structure/Union member

**wReserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_DigitalIO`  
Bases: `_ctypes.Structure`

**wDigOPs**  
Structure/Union member

**wReserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.SC_CycleParameters`  
Bases: `_ctypes.Structure`

**closedTime**  
Structure/Union member

**numCycles**  
Structure/Union member

**openTime**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KSC_MMIParams`  
Bases: `_ctypes.Structure`

**DisplayDimIntensity**  
Structure/Union member

**DisplayIntensity**  
Structure/Union member

**DisplayTimeout**  
Structure/Union member

**reserved**  
Structure/Union member

**unused**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KSC_TriggerConfig`  
Bases: `_ctypes.Structure`

**Trigger1Mode**  
Structure/Union member

**Trigger1Polarity**  
Structure/Union member

**Trigger2Mode**  
Structure/Union member

**Trigger2Polarity**  
Structure/Union member

**reserved**  
Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TIM_DriveOPParameters`  
Bases: `_ctypes.Structure`

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TIM_JogParameters`  
Bases: `_ctypes.Structure`

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TIM_ButtonParameters`  
Bases: `_ctypes.Structure`

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TIM_Status`  
Bases: `_ctypes.Structure`

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TSG_IOSettings`  
Bases: `_ctypes.Structure`

**displayMode**  
Structure/Union member

**forceCalibration**  
Structure/Union member

**futureUse**  
Structure/Union member

**hubAnalogOutput**  
Structure/Union member

**notYetInUse**  
Structure/Union member

```

class msl.equipment.resources.thorlabs.kinesis.structs.TC_LoopParameters
    Bases: _ctypes.Structure

    differentialGain
        Structure/Union member

    integralGain
        Structure/Union member

    proportionalGain
        Structure/Union member

```

## Submodules

### msl.equipment.resources.thorlabs.fw102c module

Wrapper around Thorlabs **FilterWheel102.dll**, v4.0.0.

Thorlabs FW102C Series and FW212C Series Motorized Filter Wheels.

```

class msl.equipment.resources.thorlabs.fw102c.FilterCount
    Bases: enum.IntEnum

```

The number of filter positions that the filter wheel has.

**SIX = 6**

**TWELVE = 12**

```

class msl.equipment.resources.thorlabs.fw102c.SensorMode
    Bases: enum.IntEnum

```

Sensor modes of the filter wheel.

**ON = 0**

**OFF = 1**

```

class msl.equipment.resources.thorlabs.fw102c.SpeedMode
    Bases: enum.IntEnum

```

Speed modes of the filter wheel.

**SLOW = 0**

**FAST = 1**

```

class msl.equipment.resources.thorlabs.fw102c.TriggerMode
    Bases: enum.IntEnum

```

Trigger modes of the filter wheel.

**INPUT = 0**

Respond to an active-low pulse by advancing the position by 1

**OUTPUT = 1**

Generate an active-high pulse when the position changes

```

class msl.equipment.resources.thorlabs.fw102c.FilterWheel102C(record)
    Bases: msl.equipment.connection_msl.ConnectionSDK

```

Wrapper around Thorlabs **FilterWheel102.dll**, v4.0.0.

A 64-bit version of the library can be download from [here](#) and it is located in **App-Notes\_FW102C/LabVIEW/Thorlabs\_FW102C/Library/FilterWheel102\_win64.dll**.

The `record.connection.properties` dictionary for a FilterWheel102C device supports the following key-value pairs:

```
'port': str, # mandatory, example 'COM3'  
'baud_rate': int, # optional, default is 115200  
'timeout': int, # optional, default is 10
```

**Parameters record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**Raises** `ConnectionError` – If a connection to the filter wheel cannot be established.

**close()**

Close the opened COM port.

**disconnect()**

Close the opened COM port.

**errcheck\_code** (*result, func, arguments*)

The SDK function returns OK if the function call was successful.

**errcheck\_negative** (*result, func, arguments*)

The SDK function returns a positive number if the call was successful.

**errcheck\_non\_zero** (*result, func, arguments*)

The SDK function returns 0 if the call was successful.

**get\_acceleration()**

**Returns** `int` – The current acceleration value of the filter wheel.

**get\_id()**

**Returns** `str` – The id of the filter wheel.

**get\_max\_velocity()**

**Returns** `int` – The current maximum velocity value of the filter wheel.

**get\_min\_velocity()**

**Returns** `int` – The current minimum velocity value of the filter wheel.

**get\_ports()**

List all the COM ports on the computer.

**Returns** `dict` of `str` – A dictionary where the keys are the port numbers, e.g. COM1, COM3, and the values are a description about each device connected to the port.

**get\_position()**

**Returns** `int` – The current position of the filter wheel.

**get\_position\_count()**

**Returns** *FilterCount* – The number of filter positions that the filter wheel has.

**get\_sensor\_mode()**

**Returns** *SensorMode* – The current sensor mode of the filter wheel.

**get\_speed\_mode()**

**Returns** *SpeedMode* – The current speed mode of the filter wheel.

**get\_time\_to\_current\_pos()**

**Returns** *int* – The time from last position to current position.

**get\_trigger\_mode()**

**Returns** *TriggerMode* – The current trigger mode of the filter wheel.

**is\_open(port)**

Check if the COM port is open.

**Parameters** *port* (*str*) – The port to be checked, e.g. 'COM3'.

**Returns** *bool* – *True* if the port is opened; *False* if the port is closed.

**open(port, baud\_rate, timeout)**

Open a COM port for communication.

**Parameters**

- **port** (*str*) – The port to be opened, use the *get\_ports()* function to get list of available ports.
- **baud\_rate** (*int*) – The number of bits per second to use for the communication protocol.
- **timeout** (*int*) – Set the timeout value, in seconds.

**save()**

Save the current settings as the default settings on power up.

**set\_acceleration(acceleration)**

Set the filter wheel's acceleration.

**Parameters** *acceleration* (*int*) – The filter wheel's acceleration value.

**set\_max\_velocity(maximum)**

Set the filter wheel's maximum velocity.

**Parameters** *maximum* (*int*) – The filter wheel's maximum velocity value.

**set\_min\_velocity(minimum)**

Set the filter wheel's minimum velocity.

**Parameters** *minimum* (*int*) – The filter wheel's minimum velocity value.

**set\_position(position)**

Set the filter wheel's position.

**Parameters** *position* (*int*) – The position number to set the filter wheel to.

**Raises** *ValueError* – If the value of *position* is invalid.

**set\_position\_count** (*count*)

Set the filter wheel's position count.

This is the number of filter positions that the filter wheel has.

**Parameters** **count** (*FilterCount*) – The number of filters in the filter wheel as a *FilterCount* enum value or member name.

**Raises** *ValueError* – If the value of *count* is invalid.

**set\_sensor\_mode** (*mode*)

Set the filter wheel's sensor mode.

**Parameters** **mode** (*SensorMode*) – The filter wheel's sensor mode as a *SensorMode* enum value or member name.

**Raises** *ValueError* – If the value of *mode* is invalid.

**set\_speed\_mode** (*mode*)

Set the filter wheel's speed mode.

**Parameters** **mode** (*SpeedMode*) – The speed mode of the filter wheel as a *SpeedMode* enum value or member name.

**Raises** *ValueError* – If the value of *mode* is invalid.

**set\_trigger\_mode** (*mode*)

Set the filter wheel's trigger mode.

**Parameters** **mode** (*TriggerMode*) – The filter wheel's trigger mode as a *TriggerMode* enum value or member name.

**Raises** *ValueError* – If the value of *mode* is invalid.

## Submodules

### **msl.equipment.resources.utils** module

Utility functions/classes to help create modules in the **resources** package.

`msl.equipment.resources.utils.camelcase_to_underscore` (*text*)

Converts **CamelCaseText** to **camel\_case\_text**.

**Parameters** **text** (*str*) – The camel-case text to be converted.

**Returns** *str* – The *text* converted to lowercase and separated by underscores.

`msl.equipment.resources.utils.get_lines` (*header\_path*, *move\_comments=True*) *re-*

Returns the lines in a C/C++ header file that are not empty.

Also strips the whitespace from each line and can optionally remove the all comments from the header file.

#### **Parameters**

- **header\_path** (*str*) – The path to a C/C++ header file.
- **remove\_comments** (*bool*) – Whether to remove the comments from each line.

**Returns** `list` of `str` – The list of lines in the header file.

**class** `msl.equipment.resources.utils.CHeader` (`header_path`, `re-move_comments=True`)

Bases: `object`

Parses a C/C++ header file to determine the constants, enums, structs, callbacks and the function signatures.

#### Parameters

- **header\_path** (`str`) – The path to the header file.
- **remove\_comments** (`bool`) – Whether to remove the comments from the header file.

**constants** (`ignore_ifdef=True`)

Finds the `#define` statements that in a C/C++ header file.

**Parameters** `ignore_ifdef` (`bool`, optional) – Whether to ignore the `#define` statements in between the `#ifdef` and `#endif` statements.

**Returns** `dict` of `str` – A dictionary of all the `#define` constants (as strings).

**enums** ()

**Returns** `dict` – The enums that are defined in the C/C++ header file. The value for each dictionary key is a tuple of (*the naming convention, the enum data type that is defined, a dict of name-value pairs*).

**structs** ()

**Returns** `dict` – The structs that are defined in the C/C++ header file.

**callbacks** ()

**Returns** `dict` – The callbacks that are defined in the C/C++ header file.

**functions** (`regex`)

Returns the function signatures.

**Parameters** `regex` (`str`) – The regex must create 2 groups, (*return type, function name*), and it must match the function declaration up until, but excluding, the `'('` which begins the argument declarations.

For example,

If the function declarations are similar to `FILTERFLIPPERDLL_API unsigned int __cdecl FF_GetTransitTime(const char * serialNo);` then the value of `regex` could be `r'_API\s+([\w\s]+)__cdecl\s+(\w+)'`

If the function declarations are similar to `PREF0 PREF1 PICO_STATUS PREF2 PREF3 (ps6000OpenUnit)(int16_t *handle, int8_t *serial);` then the value of `regex` could be `r'PREF0\s+PREF1\s+(\w+)\s+PREF2\s+PREF3\s+(\w+)\s+'`

**Returns** `dict` – The function signature. The key is the function name and the value is a list of [return type, [(argument data type, argument name), ... ] ].

**get\_lines** ()

**Returns** `list` of `str` – The lines in the C/C++ header file.

`get_struct_imports()`

**Returns** `list` of `str` – The list of `structs` that must be imported for the C/C++ functions.

---

**Note:** Must call `functions()` first.

---

**static** `get_text_between_brackets` (`lines`, `index`, `bracket1`, `bracket2`)

Get all the text (excluding comments) between two brackets.

**Parameters**

- **lines** (`list` of `str`) – A list of lines. Comes from `get_lines()`.
- **index** (`int`) – The current index in `lines`.
- **bracket1** (`str`) – One of {, (, or [
- **bracket2** (`str`) – One of }, ) or ]

**Returns**

- `str` – The text between `bracket1` and `bracket2`.
- `int` – The current index in `lines`.

## Submodules

### `mssl.equipment.config` module

Load a XML configuration file.

**class** `mssl.equipment.config.Config` (`path`)

Bases: `object`

Load a XML configuration file.

This function is used to set the configuration constants to use for the Python runtime and it creates `EquipmentRecord`'s from **Equipment-Register** databases and `ConnectionRecord`'s from **Connection** databases.

**MSL-Equipment** configuration constants that can be defined in a configuration file:

Name	Example Values	Description
<code>PyVISA_BACKEND</code>	@ni, @py, @sim, /path/to/lib@ni	The PyVISA <code>backend</code> library to use.
<code>DEMO_MODE</code>	True, false	Whether to open <b>all</b> connections in demo mode.
<code>SDK_PATH</code>	/path/to/SDKs, D:/data/SDKs	A path that contains SDK libraries. Accepts a <code>recursive="true"</code> attribute. Appends the path(s) to <code>os.environ['PATH']</code>

Also, the user is encouraged to define their own application-specific constants within the configuration file.



Example configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<msl>

  <!-- Application-specific constant to use as a warning if the
  ↪temperature of a device gets too hot -->
  <max_temperature units="C">60</max_temperature>

  <!-- Use PyVISA-py as the PyVISA library -->
  <PyVISA_LIBRARY>@py</PyVISA_LIBRARY>

  <!-- Open all connections in demo mode -->
  <DEMO_MODE>>true</DEMO_MODE>

  <!-- Add a path to os.environ['PATH'] where SDK files are located,
  ↪-->
  <SDK_PATH>I:\Photometry\SDKs</SDK_PATH>

  <!-- Recursively add SDK paths starting from a root path -->
  <SDK_PATH recursive="true">C:\Program Files\Thorlabs</SDK_PATH>

  <!-- The equipment that is being used to perform the measurement -
  ↪->
  <equipment alias="ref" manufacturer="Keysight" model="34465A"
  ↪serial="MY54506462"/>
  <equipment alias="scope" manufacturer="Pico Technologies" serial=
  ↪"DY135/055"/>
  <equipment alias="flipper" manufacturer="Thorlabs" model="MFF101/M
  ↪" serial="37871232"/>

  <!-- The database that contains the information required to
  ↪connect to the equipment -->
  <equipment_connections>
    <path>Z:\QUAL\Equipment\Equipment Register.xls</path>
    <sheet>Connections</sheet>
  </equipment_connections>

  <!-- The equipment-register database(s) -->
  <equipment_registers>
    <register section="P&R">
      <path>Z:\QUAL\Equipment\Equipment Register.xls</path>
      <sheet>Equipment</sheet>
    </register>
    <register section="Electrical">
      <path>H:\Quality\Registers\Equipment.xls</path>
      <sheet>REG</sheet>
    </register>
    <register section="Time">
      <path>W:\Registers\Equip.csv</path>
    </register>
    <register section="Mass">
      <path>Y:\databases\equipment\equip-reg.txt</path>
    </register>
  </equipment_registers>

</msl>
```

**Parameters** `path` (`str`) – The path to a XML configuration file.

**Raises**

- `FileNotFoundError` – If `path` does not exist.
- `ParseError` – If the configuration file is invalid.

**PyVISA\_LIBRARY = '@ni'**

`str` – The PyVISA backend library to use

**DEMO\_MODE = False**

`bool` – Whether to open **all** connections in demo mode

**SDK\_PATH = []**

`list of str` – Paths that were appended to `os.environ['PATH']`

**path**

`str` – The path to the configuration file.

**root**

Returns the root element (the first node) of the XML tree.

**Returns** `Element` – The root element.

**database ()**

**Returns** `Database` – A reference to the equipment and connection records in the database(s) that are specified in the configuration file.

**value (`tag`)**

Gets the value associated with the specified `tag`.

**Parameters** `tag` (`str`) – The name of a XML tag in the configuration file.

**Returns** `str` or `None` – The value associated with the `tag` or `None` if the tag cannot be found.

## msl.equipment.connection module

Base class for establishing a connection to the equipment.

**class** `msl.equipment.connection.Connection` (`record`)

Bases: `object`

All `Backend` Connection classes must have this class as the base class.

Do not instantiate this class directly. Use the factory method, `msl.equipment.factory.connect`, or the `record` object itself, `record.connect ()`, to connect to the equipment.

**Parameters** `record` (`EquipmentRecord`) – An equipment record from an **Equipment-Register** `Database`.

**equipment\_record**

`EquipmentRecord` – The equipment record, from an **Equipment-Register** `Database`, that is being used to establish the connection.

**disconnect ()**

Disconnect from the equipment.

This method should be overridden in the *Connection* subclass if the subclass must implement tasks that need to be performed in order to safely disconnect from the equipment.

For example:

- to clean up system resources from memory (e.g., if using a manufacturer's SDK)
- to configure the equipment to be in a state that is safe for people working in the lab when the equipment is not in use

---

**Note:** This method gets called automatically when the *Connection* object gets destroyed.

---

**raise\_exception** (*msg*)

Raise a *ConnectionError* exception.

**Parameters** *msg* (*str*) – The message to display when the exception is raised.

**static convert\_to\_enum** (*item*, *enum*, *prefix*='', *to\_upper*=*False*)

Convert *item* to an *enum* value.

**Parameters**

- **item** (*int* or *str*) – If *str* then the name of a *enum* member. If *int* then the value of a *enum* member.
- **enum** (*IntEnum*) – An integer-enum object.
- **prefix** (*str*, optional) – If *item* is a *str*, then *prefix* is included at the beginning of *item* before converting *item* to an *enum* value.
- **to\_upper** (*bool*, optional) – If *item* is a *str*, then whether to change *item* to be upper case before converting *item* to an *enum* value.

**Returns** *IntEnum* – The *enum* value.

**Raises**

- *TypeError* – If *item* is not an *int* or *str*.
- *ValueError* – If *item* is not in *enum*.

**static log\_debug** (*msg*, *\*args*, *\*\*kwargs*)

Log a *debug* message.

**Parameters** *msg* (*str*) – The debug message to log.

**static log\_info** (*msg*, *\*args*, *\*\*kwargs*)

Log an *info* message.

**Parameters** *msg* (*str*) – The info message to log.

**static log\_warning** (*msg*, *\*args*, *\*\*kwargs*)

Log a *warning* message.

**Parameters** *msg* (*str*) – The warning message to log.

**static log\_error** (*msg*, *\*args*, *\*\*kwargs*)

Log an *error* message.

**Parameters** *msg* (*str*) – The error message to log.

```
static log_critical (msg, *args, **kwargs)
    Log a critical message.
```

**Parameters** `msg` (`str`) – The critical message to log.

### `mssl.equipment.connection_demo` module

Simulate a connection to the equipment.

```
class mssl.equipment.connection_demo.ConnectionDemo (record, cls)
    Bases: mssl.equipment.connection.Connection
```

Simulate a connection to the equipment.

Establishing a connection in demo mode is useful when developing a program and the equipment is not physically connected to the computer.

A custom `logging level` is used for logging messages with a connection in demo mode. The `logging.DEMO logging level` is set to be between `logging.INFO` and `logging.WARNING`.

The returned data type is determined from the docstring of the called function. For example, if `:rtype: int` then an integer value is returned or if `:rtype: int, float` then an integer and a float value is returned. Although the expected data type is returned the value(s) of the returned object is randomly generated. The docstring must be in either the `reStructuredText` or `NumPy` format.

Do not instantiate this class directly. Use the factory method, `mssl.equipment.factory.connect`, or the `record` object itself, `record.connect()`, to connect to the equipment in demo mode.

#### Parameters

- **record** (`EquipmentRecord`) – An equipment record from an **Equipment-Register Database**.
- **cls** (`Connection`) – A `Connection` class (that has **NOT** been instantiated).

```
disconnect ()
```

Log a disconnection from the equipment.

### `mssl.equipment.connection_mssl` module

Use MSL resources to establish a connection to the equipment.

```
class mssl.equipment.connection_mssl.ConnectionSDK (record, libtype)
    Bases: mssl.equipment.connection.Connection
```

Base class for equipment that use the SDK provided by the manufacturer for the connection.

The `record.connection.backend` value must be equal to `Backend.MSL` to use this class for the communication system. This is achieved by setting the value in the **Backend** field for a connection record in the **Connections** database to be **MSL**.

Do not instantiate this class directly. Use the factory method, `mssl.equipment.factory.connect`, or the `record` object itself, `record.connect()`, to connect to the equipment.

#### Parameters

- **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.
- **libtype** (*str* or *None*) – The library type to use for the calling convention.  
The following values are allowed:
  - ‘**cdll**’ – for a **\_\_cdecl** library
  - ‘**windll**’ or ‘**oledll**’ – for a **\_\_stdcall** library (Windows only)
  - ‘**net**’ – for a **.NET** library

**Raises** *IOError* – If the Python wrapper class around the SDK cannot be found or if the shared library cannot be found.

**sdk\_path**

*str* – The path to the shared library file.

**sdk**

The reference to the *sdk* library.

**log\_errcheck** (*result, func, arguments*)

Convenience method for logging an *errcheck* when calling an SDK function.

**class** *msl.equipment.connection\_msl.ConnectionMessageBased* (*record*)

Bases: *msl.equipment.connection.Connection*

Base class for equipment that use message based communication.

The *record.connection.backend* value must be equal to *Backend.MSL* to use this class for the communication system. This is achieved by setting the value in the **Backend** field for a connection record in the **Connections** database to be **MSL**.

Do not instantiate this class directly. Use the factory method, *msl.equipment.factory.connect*, or the *record* object itself, *record.connect()*, to connect to the equipment.

**Parameters record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**CR** = ‘\r’

*str* – The carriage-return character

**LF** = ‘\n’

*str* – The line-feed character

**encoding**

*str* – The encoding that is used for *read()* and *write()* operations.

**read\_termination**

*str* or *None* – The termination character sequence that is used for *read()* operations.

Reading stops when the equipment stops sending data (e.g., by setting appropriate bus lines) or the *read\_termination* character sequence is detected.

**write\_termination**

*str* – The termination character sequence that is appended to *write()* messages.

**chunk\_size**

*int* – The number of bytes to be *read()*.

**read\_size**

`int` – The number of bytes to be `read()`.

**read** (*size=None*)

Read a response from the equipment.

**Returns** `str` – The response from the equipment.

**write** (*message*)

Write (send) a message to the equipment.

**Parameters** `message` (`str`) – The message to write (send) to the equipment.

**Returns** `int` – The number of bytes written.

**send** (*message*)

Write (send) a message to the equipment.

**Parameters** `message` (`str`) – The message to write (send) to the equipment.

**Returns** `int` – The number of bytes written.

**query** (*message, delay=0.0*)

Convenience method for performing a `write()` followed by a `read()`.

**Parameters**

- **message** (`str`) – The message to write (send) to the equipment.
- **delay** (`float`) – The time delay, in seconds, to wait between `write()` and `read()` operations.

**Returns** `str` – The response from the equipment.

**ask** (*message, delay=0.0*)

Convenience method for performing a `write()` followed by a `read()`.

**Parameters**

- **message** (`str`) – The message to write (send) to the equipment.
- **delay** (`float`) – The time delay, in seconds, to wait between `write()` and `read()` operations.

**Returns** `str` – The response from the equipment.

**class** `msl.equipment.connection_msl.ConnectionSerial` (*record*)

Bases: `msl.equipment.connection_msl.ConnectionMessageBased`

Establish a connection through a Serial port.

The `record.connection.properties` dictionary for a Serial connection supports the following key-value pairs:

```
'read_termination': str or None
'write_termination': str or None
'read_size': int (the number of bytes to be read, must be > 0)
'encoding': str (e.g., 'ascii')
'baud_rate': int (e.g., 9600, 115200)
'data_bits': int (e.g., 5, 6, 7, 8)
'stop_bits': int or float (e.g., 1, 1.5, 2)
'parity': str (e.g., none, even, odd, mark, space)
'timeout': float or None (the read timeout value)
```

```
'write_timeout': float or None (the write timeout value)
'inter_byte_timeout': float or None (the inter-character timeout)
'exclusive': bool (set exclusive access mode, for POSIX only)
'xon_xoff': bool (enable software flow control)
'rts_cts': bool (enable hardware (RTS/CTS) flow control)
'dsr_dtr': bool (enable hardware (DSR/DTR) flow control)
```

**serial**

`serial.Serial` – The reference to the Serial object.

**baud\_rate**

`int` – The baud rate setting.

**data\_bits**

`DataBits` – The number of data bits.

**stop\_bits**

`StopBits` – The stop bit setting.

**parity**

`Parity` – The parity setting.

**write** (*message*)

Write the given message over the serial port.

**Parameters** `message` (`str`) – The message to write.

**Returns** `int` – The number of bytes written to the serial port.

**read** (*size=None*)

Read *size* bytes from the serial port.

If a *timeout* is set it may return less characters as requested. With no *timeout* it will block until the requested number of bytes is read.

**Parameters** `size` (`int`) – The number of bytes to read. If `None` then read `read_size` bytes.

**Returns** `bytes` – The bytes read from the serial port.

**disconnect** ()**msl.equipment.connection\_pyvisa module**

Use `PyVISA` as the backend to communicate with the equipment.

**class** `msl.equipment.connection_pyvisa.ConnectionPyVISA` (*record*)

Bases: `msl.equipment.connection.Connection`

Use `PyVISA` to establish a connection to the equipment.

The `record.connection.backend` value must be equal to `Backend.PyVISA` to use this class for the communication system. This is achieved by setting the value in the **Backend** field for a connection record in the **Connections** database to be **PyVISA**.

If you want to change the `read_termination`, `write_termination` and/or the encoding value for communication with the equipment then you can define, for exam-

ple, `read_termination=\n`; `write_termination=\n`; `encoding=utf-8` in the **Properties** field for a connection record in the **Connections** database.

Do not instantiate this class directly. Use the factory method, `mssl.equipment.factory.connect`, or the `record` object itself, `record.connect()`, to connect to the equipment.

**Parameters** `record` (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

**disconnect** ()

Calls `close()`.

**static resource\_manager** (*visa\_library=None*)

Return the PyVISA `ResourceManager`.

Only **one** Resource Manager session is created per Python runtime and therefore multiple calls to this function will return the same `ResourceManager` object.

**Parameters** `visa_library` (*VisaLibraryBase, str or None*) – The library to use for PyVISA. For example:

- `@ni` to use `NI-VISA`
- `@py` to use `PyVISA-py`
- `@sim` to use `PyVISA-sim`

If `None` then the `visa_library` value is read from a `CONFIG` variable. See `mssl.equipment.config.load` for more details.

**Returns** `ResourceManager` – The PyVISA Resource Manager.

**Raises**

- `ValueError` – If the PyVISA backend wrapper cannot be found.
- `OSError` – If the VISA library cannot be found.

**static resource\_pyclass** (*record*)

Find the PyVISA `Resource` that can open the `record`.

**Parameters** `record` (*EquipmentRecord or ConnectionRecord*) – An equipment or connection record from the *Database*.

**Returns** `Resource` – The appropriate PyVISA Resource class that can open the `record`.

### **mssl.equipment.constants module**

MSL-Equipment package constants.

**class** `mssl.equipment.constants.Backend`

Bases: `enum.IntEnum`

The software backend to use for the communication system.

**UNKNOWN = 0**

**MSL = 1**

**PyVISA = 2**



**class** `msl.equipment.constants.MSLInterface`

Bases: `enum.IntEnum`

The interface to use for the communication system that transfers data between a computer and the equipment. Only used if `Backend.MSL` is chosen as the communication system.

**NONE = 0**

**ASRL = 1**

**GPIB = 2**

**USB = 3**

**PXI = 4**

**VXI = 5**

**TCPIP = 6**

**SDK = 7**

**TCPIP\_ASRL = 8**

**TCPIP\_GPIB = 9**

**USB\_ASRL = 10**

**USB\_GPIB = 11**

**PROLOGIX\_ENET = 12**

**PROLOGIX\_USB = 13**

**class** `msl.equipment.constants.Parity`

Bases: `enum.Enum`

The parity type to use for Serial communication.

**NONE = 'N'**

**ODD = 'O'**

**EVEN = 'E'**

**MARK = 'M'**

**SPACE = 'S'**

**class** `msl.equipment.constants.StopBits`

Bases: `enum.Enum`

The number of stop bits to use for Serial communication.

**ONE = 1**

**ONE\_POINT\_FIVE = 1.5**

**TWO = 2**

**class** `msl.equipment.constants.DataBits`

Bases: `enum.IntEnum`

The number of data bits to use for Serial communication.

**FIVE = 5**

**SIX = 6**

**SEVEN = 7**

**EIGHT = 8**

### msl.equipment.database module

Load equipment and connection records (rows) from databases.

**class** `msl.equipment.database.Database` (*path*)

Bases: `object`

Create *EquipmentRecord*'s and *ConnectionRecord*'s from databases that are specified in a configuration file.

See `msl.equipment.config.load` for an example of a XML configuration file.

**Parameters** *path* (`str`) – The path to a XML configuration file.

#### Raises

- `FileNotFoundError` – If *path* does not exist.
- `ParseError` – If the configuration file is invalid.
- `IOError` – If an XML element content is invalid.
- `AttributeError` – If an `<equipment>` XML element is specified in the configuration file and it does not uniquely identify an equipment record in an **Equipment-Register** database.
- `ValueError` – If any of the values in the **Connections** database are invalid.

#### equipment

`dict` of *EquipmentRecord* – Equipment records that were listed as `<equipment>` XML elements in the configuration.

#### path

`str` – The path to the configuration file that contains the information about the **Equipment-Register** and **Connections** databases and the environment variables.

#### connections (\*\*kwargs)

Search the **Connections** database to find all connection records that match the specified criteria.

**Parameters** **\*\*kwargs** – The argument names can be any of the *ConnectionRecord* property names. The comparison for the value is performed by `regex`, see `re`.

**Returns** `list` of *ConnectionRecord* – Connection records that match the search criteria.

### Examples

```
>>> connections()
a list of all ConnectionRecords
```

```
>>> connections(manufacturer='H*P')
a list of all ConnectionRecords that have Hewlett Packard as the
↳ manufacturer
```

```
>>> connections(address='GPIB*')
a list of all ConnectionRecords that use GPIB for the connection
↳ bus
```

### **records** (\*\*kwargs)

Search the **Equipment-Register** databases to find all equipment records that match the specified criteria.

**Parameters** **\*\*kwargs** – The argument names can be any of the *EquipmentRecord* property names. The comparison for the value is performed by *regex*, see *re*.

**Returns** *list of EquipmentRecord* – Equipment records that match the search criteria.

### Examples

```
>>> records()
a list of all EquipmentRecords
```

```
>>> records(manufacturer='H*P')
a list of all EquipmentRecords that have Hewlett Packard as the
↳ manufacturer
```

```
>>> records(manufacturer='Agilent', model='3458A')
a list of all EquipmentRecords that are from Agilent and that
↳ have the model number 3458A
```

```
>>> records(manufacturer='Agilent', model='3458A', serial=
↳ 'MY45046470')
a list of only one EquipmentRecord (if the equipment record
↳ exists, otherwise an empty list)
```

```
>>> records(description='I-V Converter')
a list of all EquipmentRecords that contain 'I-V Converter' in
↳ the description field
```

## msl.equipment.factory module

Establish a connection to the equipment to send and receive messages.

`msl.equipment.factory.connect` (*record*, *demo=None*)

Factory function to establish a connection to the equipment.

### Parameters

- **record** (*EquipmentRecord*) – An equipment record from an **Equipment-Register Database**.

- **demo** (`bool` or `None`) – Whether to simulate a connection to the equipment by opening a connection in demo mode. This allows you run your code if the equipment is not physically connected to the computer.

If `None` then the `demo` value is read from a configuration variable. See `msl.equipment.config.Config` for more details.

**Returns** `Connection` – A `Connection`-type object.

**Raises** `ValueError` – If any of the property values in `record.connection.properties` are invalid.

## msl.equipment.record\_types module

A record from an **Equipment-Register** database or a **Connections** database.

**class** `msl.equipment.record_types.EquipmentRecord` (`**kwargs`)

Bases: `object`

Contains the information about an equipment record in an **Equipment-Register** database.

**Parameters** `**kwargs` – The argument names can be any of the `EquipmentRecord` attribute names.

**Raises**

- `ValueError` – If an argument name is `calibration_period`, `connection` or `date_calibrated` and the value is invalid.
- `AttributeError` – If a named argument is not an `EquipmentRecord` attribute name.

## Examples

Equipment records **should** be defined in a database and instantiated by calling `msl.equipment.config.load`; however, you can still manually create an equipment record.

```
>>> from msl.equipment import EquipmentRecord, ConnectionRecord, Backend
... record = EquipmentRecord(
...     manufacturer='Pico Technology',
...     model='5244B',
...     serial='DY135/055',
...     connection=ConnectionRecord(
...         backend=Backend.MSL,
...         address='SDK::PicoScope5000A::ps5000a',
...         properties={
...             'resolution': '14bit', # only used for
...             'auto_select_power': True, # for PicoScopes
...         },
...     )
... )
```

**alias**

`str` – An alias to use to associate with this equipment.

**asset\_number**

`str` – The IRL/CI asset number of the equipment.

**calibration\_period**

`int` – The number of years that can pass before the equipment must be recalibrated.

**category**

`str` – The category (e.g., Laser, DMM) that the equipment belongs to.

**connection**

*ConnectionRecord* – The information necessary to establish a connection to the equipment.

**date\_calibrated**

`datetime.date` – The date that the equipment was last calibrated.

**description**

`str` – A description of the equipment.

**location**

`str` – The location where the equipment can usually be found.

**manufacturer**

`str` – The name of the manufacturer of the equipment.

**model**

`str` – The model number of the equipment.

**register**

`str` – The value assigned, as in MSL Policy and Procedures, for any equipment that requires calibration or maintenance for projects.

**section**

`str` – The MSL section (e.g., P&R) that the equipment belongs to.

**serial**

`str` – The serial number, or engraved unique ID, of the equipment.

**static attributes ()**

`list of str`: A list of all the attribute names for an *EquipmentRecord* object.

**connect (demo=None)**

Establish a connection to the equipment.

**Parameters** `demo` (`bool` or `None`) – Whether to simulate a connection to the equipment by opening a connection in demo mode. This allows you run your code if the equipment is not physically connected to the computer.

If `None` then the `demo` value is read from a `CONFIG` variable. See `msl.equipment.config.load` for more details.

**Returns** *Connection* – A *Connection*-type object.

**Raises** *ValueError* – If any of the property values in `record.connection.properties` are invalid.

```
class msl.equipment.record_types.ConnectionRecord (**kwargs)
    Bases: object
```

Contains the information about a connection record in a **Connections** database.

**Parameters `**kwargs`** – The argument names can be any of the `ConnectionRecord` attribute names.

### Raises

- `ValueError` – If an argument name is `backend`, `interface` or `properties` and the value is invalid.
- `AttributeError` – If a named argument is not an `ConnectionRecord` attribute name.

### Examples

Connection records **should** be defined in a database and instantiated by calling `msl.equipment.config.load`; however, you can still manually create a connection record.

```
>>> from msl.equipment import ConnectionRecord, Backend
>>> record = ConnectionRecord(
...     manufacturer='Pico Technology',
...     model='5244B',
...     serial='DY135/055',
...     backend=Backend.MSL,
...     address='SDK::PicoScope5000A::ps5000a',
...     properties={
...         'resolution': '14bit', # only used for ps5000a
↪series PicoScope's
...         'auto_select_power': True, # for PicoScopes
↪that can be powered by an AC adaptor or by a USB cable
...     }
... )
```

### address

`str` – The address to use for the connection (see [here](#) for examples from National Instruments).

### backend

`Backend` – The backend to use to communicate with the equipment.

### interface

`MSLInterface` – The interface to use for the communication system that transfers data between a computer and the equipment (only used if the `backend` is `Backend.MSL`).

### manufacturer

`str` – The name of the manufacturer of the equipment.

### model

`str` – The model number of the equipment.

### properties

`dict` – Additional properties that may be used to establish a connection to the equipment, e.g., for a Serial connection `{'baud_rate': 11920, 'data_bits': 8}`.

### serial

`str` – The serial number, or engraved unique ID, of the equipment.

**static attributes** ()

`list of str`: A list of all the attribute names for a *ConnectionRecord* object.





---

## MSL-Equipment Database Formats

---

Databases are used by **MSL-Equipment** to store *EquipmentRecord*'s in an *Equipment-Register Database* and *ConnectionRecord*'s in a *Connections Database*. The database file formats that are currently supported are **.txt**, **.csv** and **.xls(x)**. A database is made up of **fields** (columns) and **records** (rows).

### Equipment-Register Database

To adhere to international ISO/IEC 17025 standards, the information about the equipment that is used for a calibration or for a comparison must be known and it must be up to date. Keeping a central database of the equipment that is available in the lab allows for managing this information and for helping to ensure that the equipment that is being used for a calibration/comparison meets the requirements needed to achieve the final measurement uncertainty.

**MSL-Equipment** does not require that a *single* database is used for **ALL** equipment from **ALL** MSL sections. However, it is vital that each equipment record can only be uniquely found in **ONE Equipment-Register** database. These databases are never to be copied from one MSL section to another (*although keeping backups are required*). Rather if you are borrowing equipment from another team you simply specify the path to that team's **Equipment-Register** database in your configuration file. See *mssl.equipment.config.Config* for more details on how to specify multiple **Equipment-Register** databases in a configuration file. The owner of the equipment is responsible for ensuring that the information about the equipment is updated in their **Equipment-Register** database. The user of the equipment has access to this information through an *EquipmentRecord* object.

Each **record** in an **Equipment-Register** database is converted into an *EquipmentRecord* object when a configuration file is loaded. Each **field** is an property name for an *EquipmentRecord*.

Example **Equipment-Register** database:

<p>The Register number used for Policies and Procedures</p>	<p>A column header can be used any text. The name of an <i>EquipmentRecord</i> property must appear somewhere in the header text to associate the values in this column with the <i>EquipmentRecord</i> property value. This column is used to specify the Manufacturer</p>	<p>Model #</p>
<p>198</p>	<p>turer of the equip-</p>	<p>Chapter 4. MSL-Equipment Database Formats</p>

# Connections Database

Describe



MSL resources that are available for establishing a connection to equipment.

### SDK Wrapper Classes

The following classes, which are a wrapper around the Software Development Kit (SDK) provided by the manufacturer, are available to use when defining the **Address** field in a **Connections** database.

The format of an **Address** is `SDK::PythonClassName::PathToLibrary` where `PathToLibrary` is the full path to the SDK file (or only the filename if the library path is available on `os.environ['PATH']`, see *Config* for more details) and `PythonClassName` can be one of the following classes.

For example, the **Address** field for equipment from **Bentham Instruments Ltd** can be `SDK::Bentham::benhw32_cdecl.dll`.

- Bentham Instruments Ltd
  - *Bentham* - The `benhw32` library
- Pico Technology
  - *PicoScope*
    - \* *PicoScope2000* - ps2000
    - \* *PicoScope2000A* - ps2000a
    - \* *PicoScope3000* - ps3000
    - \* *PicoScope3000A* - ps3000a
    - \* *PicoScope4000* - ps4000
    - \* *PicoScope4000A* - ps4000a
    - \* *PicoScope5000* - ps5000

\* *PicoScope5000A* - ps5000a

\* *PicoScope6000* - ps6000

- Thorlabs

- *Kinesis* - Wrapper package around `Thorlabs.MotionControl.C_API`.

- \* *FilterFlipper* - MFF101, MFF102

- \* *IntegratedStepperMotors* - LTS150, LTS300, MLJ050, K10CR1

- \* *KCubeSolenoid* - KSC101

- *FilterWheel102C* - FW102C, FW212C

## CHAPTER 6

---

### License

---

MIT License

Copyright (c) 2017, Joseph Borbely

Permission **is** hereby granted, free of charge, to **any** person obtaining a **copy** of this software **and** associated documentation files (the "**Software**"), to **deal** **in** the Software without restriction, including without limitation the **rights** to use, copy, modify, merge, publish, distribute, sublicense, **and/or** sell copies of the Software, **and** to permit persons to whom the Software **is** furnished to do so, subject to the following conditions:

The above copyright notice **and** this permission notice shall be included **in** **all** copies **or** substantial portions of the Software.

THE SOFTWARE IS PROVIDED "**AS IS**", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING **FROM**, **OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN** **THE SOFTWARE**.





## CHAPTER 7

---

Developers

---

- Joseph Borbely <[joseph.borbely@measurement.govt.nz](mailto:joseph.borbely@measurement.govt.nz)>



## CHAPTER 8

---

### Release Notes

---

#### **Version 0.1.0**

- Initial release



## CHAPTER 9

---

### Index

---

- modindex



---

## Python Module Index

---

### m

`mssl.equipment`, 7

`mssl.equipment.config`, 180

`mssl.equipment.connection`, 182

`mssl.equipment.connection_demo`, 184

`mssl.equipment.connection_mssl`, 184

`mssl.equipment.connection_pyvisa`, 187

`mssl.equipment.constants`, 188

`mssl.equipment.database`, 190

`mssl.equipment.factory`, 191

`mssl.equipment.record_types`, 192

`mssl.equipment.resources`, 8

`mssl.equipment.resources.bentham`, 9

`mssl.equipment.resources.bentham.bentham32`, 9

`mssl.equipment.resources.bentham.bentham64`, 11

`mssl.equipment.resources.bentham.errors`, 12

`mssl.equipment.resources.bentham.tokens`, 12

`mssl.equipment.resources.picotech`, 12

`mssl.equipment.resources.picotech.picoscope`, 12

`mssl.equipment.resources.picotech.picoscope.callbacks`, 12

`mssl.equipment.resources.picotech.picoscope.channel`, 12

`mssl.equipment.resources.picotech.picoscope.enums`, 14

`mssl.equipment.resources.picotech.picoscope.errors`, 64

`mssl.equipment.resources.picotech.picoscope.functions`, 64

`mssl.equipment.resources.picotech.picoscope.helper`, 64

`mssl.equipment.resources.picotech.picoscope.picotech.picoscope.pi`, 65

`mssl.equipment.resources.picotech.picoscope.pi`, 69

`mssl.equipment.resources.picotech.picoscope.pi`, 71

`mssl.equipment.resources.picotech.picoscope.ps`, 80

`mssl.equipment.resources.picotech.picoscope.ps`, 82

`mssl.equipment.resources.picotech.picoscope.ps`, 83

`mssl.equipment.resources.picotech.picoscope.ps`, 85

`mssl.equipment.resources.picotech.picoscope.ps`, 87

`mssl.equipment.resources.picotech.picoscope.ps`, 89

`mssl.equipment.resources.picotech.picoscope.ps`, 90

`mssl.equipment.resources.picotech.picoscope.ps`, 91

`mssl.equipment.resources.picotech.picoscope.ps`, 93

`mssl.equipment.resources.picotech.picoscope.st`, 94

`mssl.equipment.resources.thorlabs`, 106

`mssl.equipment.resources.thorlabs.fw102c`, 175

`mssl.equipment.resources.thorlabs.kinesis`, 106

`mssl.equipment.resources.thorlabs.kinesis.api`, 106

`mssl.equipment.resources.thorlabs.kinesis.call`, 106

`mssl.equipment.resources.thorlabs.kinesis.enum`, 107

`mssl.equipment.resources.thorlabs.kinesis.error`

123  
msl.equipment.resources.thorlabs.kinesis.filter\_flipper,  
123  
msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors,  
127  
msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid,  
146  
msl.equipment.resources.thorlabs.kinesis.messages,  
153  
msl.equipment.resources.thorlabs.kinesis.motion\_control,  
154  
msl.equipment.resources.thorlabs.kinesis.structs,  
157  
msl.equipment.resources.utils, 178



**A**

- A (msl.equipment.resources.picotech.picoscope.enums.PS2000Channel attribute), 18
- A (msl.equipment.resources.picotech.picoscope.enums.PS2000Channel attribute), 14
- A (msl.equipment.resources.picotech.picoscope.enums.PS3000Channel attribute), 28
- A (msl.equipment.resources.picotech.picoscope.enums.PS3000Channel attribute), 24
- A (msl.equipment.resources.picotech.picoscope.enums.PS4000Channel attribute), 41
- A (msl.equipment.resources.picotech.picoscope.enums.PS4000Channel attribute), 35
- A (msl.equipment.resources.picotech.picoscope.enums.PS5000Channel attribute), 55
- A (msl.equipment.resources.picotech.picoscope.enums.PS5000Channel attribute), 50
- A (msl.equipment.resources.picotech.picoscope.enums.PS6000Channel attribute), 59
- A\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000ChannelBufferIndex attribute), 18
- A\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ChannelBufferIndex attribute), 28
- A\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 41
- A\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 34
- A\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 60
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS2000ChannelBufferIndex attribute), 18
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS3000ChannelBufferIndex attribute), 28
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 41
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 34
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 60
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS2000Channel attribute), 34
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS3000Channel attribute), 50
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000Channel attribute), 60
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000Channel attribute), 23
- A\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS6000Channel attribute), 45
- ABOVE (msl.equipment.resources.picotech.picoscope.enums.PS2000Channel attribute), 17
- ABOVE (msl.equipment.resources.picotech.picoscope.enums.PS3000Channel attribute), 33
- ABOVE (msl.equipment.resources.picotech.picoscope.enums.PS4000Channel attribute), 45
- ABOVE (msl.equipment.resources.picotech.picoscope.enums.PS5000Channel attribute), 58
- ABOVE (msl.equipment.resources.picotech.picoscope.enums.PS6000Channel attribute), 63
- ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS2000ChannelBufferIndex attribute), 23
- ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS3000ChannelBufferIndex attribute), 33
- ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 45
- ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 39
- ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 63

ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000.ATHreshholdDirection attribute), 58  
 ABOVE\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS6000.ATHreshholdDirection attribute), 63  
 absoluteReading (msl.equipment.resources.thorlabs.kinesis.structs.NT\_TipTheReading attribute), 164  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS2000.ACOutputHz50MV attribute), 20  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS3000.ACOutputHz attribute), 30  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS4000.ACOutput attribute), 41  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS5000.ACOutputHz5V attribute), 54  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS6000.ACOutputHz attribute), 61  
 ACCELERATION (msl.equipment.resources.thorlabs.kinesis.structs.MDEquipmentRecordType attribute), 127  
 acceleration (msl.equipment.resources.thorlabs.kinesis.structs.MDEquipmentRecordType.ConnectionRecord attribute), 158  
 accelerationFeedForward (msl.equipment.resources.thorlabs.kinesis.structs.MDEquipmentRecordType attribute), 160  
 ACCELEROMETER (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 37  
 ACCELEROMETER\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 36  
 ACCELEROMETER\_100V (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 36  
 ACCELEROMETER\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 35  
 ACCELEROMETER\_10V (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 36  
 ACCELEROMETER\_1V (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 36  
 ACCELEROMETER\_200MV (msl.equipment.resources.picotech.picoscope.enums.PS1000.ACOutput attribute), 36  
 ACCELEROMETER\_20MV (msl.equipment.resources.picotech.picoscope.enums.PS4000.ACOutput attribute), 35  
 ACCELEROMETER\_20V AllowAllMoves (msl.equipment.resources.thorlabs.kinesis.enums.M

attribute), 109

AllowPartialMoves (msl.equipment.resources.thorlabs.kinesis.enums.MSLEquipmentSoftwareApproachPolicy attribute), 109

amplifierCurrentLimit (msl.equipment.resources.thorlabs.kinesis.structs.BNAutoSettings attribute), 163

amplifierLowPassFilter (msl.equipment.resources.thorlabs.kinesis.structs.BNAutoSettings attribute), 163

ANALOGUE\_HARDWARE\_VERSION (msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfoApi attribute), 14

AnalogueCh1 (msl.equipment.resources.thorlabs.kinesis.enums.MSLHardwareAnalogueModes attribute), 119

AnalogueCh2 (msl.equipment.resources.thorlabs.kinesis.enums.MSLHardwareAnalogueModes attribute), 119

AND (msl.equipment.resources.picotech.picoscope.enums.PS2000ATriggerOperand attribute), 19

anticlockwiseHardwareLimit (msl.equipment.resources.thorlabs.kinesis.structs.MSLEquipmentSwitchParameters attribute), 162

anticlockwisePosition (msl.equipment.resources.thorlabs.kinesis.structs.MSLEquipmentSwitchParameters attribute), 162

apply\_resistance\_scaling() (msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerParameters attribute), 90

ARM (msl.equipment.resources.picotech.picoscope.enums.PS5000ATriggerWindowPicoTrigger attribute), 58

ARRAY\_OF\_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS4000AProbeStringValue attribute), 47

ask() (msl.equipment.connection\_msl.ConnectionMessageBased attribute), 186

ASRL (msl.equipment.constants.MSLInterface attribute), 189

asset\_number (msl.equipment.record\_types.EquipmentRecord attribute), 193

attributes() (msl.equipment.record\_types.ConnectionRecord attribute), 194

attributes() (msl.equipment.record\_types.EquipmentRecord attribute), 193

auto\_measure() (msl.equipment.resources.bentham.benhw32.Bentham attribute), 9

auto\_measure() (msl.equipment.resources.bentham.benhw64.Bentham attribute), 11

auto\_range() (msl.equipment.resources.bentham.benhw32.Bentham attribute), 9

AUX (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel attribute), 18

AUX (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel attribute), 28

AUX (MSL Equipment Software Approach Policy attribute), 41

AUX (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel attribute), 55

AUX (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 50

AUX (msl.equipment.resources.picotech.picoscope.enums.PS6000AChannel attribute), 10

aux (msl.equipment.resources.picotech.picoscope.structs.PS2000AChannel attribute), 10

aux (msl.equipment.resources.picotech.picoscope.structs.PS2000AChannel attribute), 10

aux (msl.equipment.resources.picotech.picoscope.structs.PS3000AChannel attribute), 10

aux (msl.equipment.resources.picotech.picoscope.structs.PS3000AChannel attribute), 98

aux (msl.equipment.resources.picotech.picoscope.structs.PS3000AChannel attribute), 98

aux (msl.equipment.resources.picotech.picoscope.structs.PS4000Probe attribute), 101

aux (msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerParameters attribute), 104

aux (msl.equipment.resources.picotech.picoscope.structs.PS5000Probe attribute), 103

aux (msl.equipment.resources.picotech.picoscope.structs.PS5000Probe attribute), 102

aux (msl.equipment.resources.picotech.picoscope.structs.PS6000Probe attribute), 105

AUX\_IN (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel attribute), 22

AUX\_IN (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel attribute), 22

AUX\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel attribute), 38

AUX\_IN (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 38

AUX\_IN (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 38

attribute), 52  
 AUX\_IN (msl.equipment.resources.picotech.picoscope.enums.PS6000SigGenTrigSource attribute), 62  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS2000ARatioMode attribute), 24  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS3000ARatioMode attribute), 34  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode attribute), 46  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS4000ERatioMode attribute), 40  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode attribute), 59  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS5000ERatioMode attribute), 53  
 AVERAGE (msl.equipment.resources.picotech.picoscope.enums.PS6000ARatioMode attribute), 64  
 AWG\_DAC\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps4000aPiezoSine4000a resources.picotech.picoscope.enums.PS4000 attribute), 89  
 AWG\_DAC\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000aPiezoSine5000a resources.picotech.picoscope.ps5000aPiezoSine5000a attribute), 92  
 AWG\_PHASE\_ACCUMULATOR (msl.equipment.resources.picotech.picoscope.ps4000aPiezoSine4000a resources.picotech.picoscope.enums.PS5000 attribute), 89  
 AWG\_PHASE\_ACCUMULATOR (msl.equipment.resources.picotech.picoscope.ps5000aPiezoSine5000a resources.picotech.picoscope.ps5000aPiezoSine5000a attribute), 92  
 axisID (msl.equipment.resources.thorlabs.kinesis.structs.MOTInStageAxisParameters types.ConnectionRecord attribute), 159  
**B**  
 B (msl.equipment.resources.picotech.picoscope.enums.PS2000ANDaSERIAL attribute), 18  
 B (msl.equipment.resources.picotech.picoscope.enums.PS2000GAttribute), 14  
 B (msl.equipment.resources.picotech.picoscope.enums.PS3000AAttribute), 28  
 B (msl.equipment.resources.picotech.picoscope.enums.PS3000ANDaSERIAL attribute), 24  
 B (msl.equipment.resources.picotech.picoscope.enums.PS4000AAttribute), 41  
 B (msl.equipment.resources.picotech.picoscope.enums.PS4000GAttribute), 35  
 B (msl.equipment.resources.picotech.picoscope.enums.PS5000AAttribute), 55  
 B (msl.equipment.resources.picotech.picoscope.enums.PS5000GAttribute), 50  
 B (msl.equipment.resources.picotech.picoscope.enums.PS6000AAttribute), 59  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000ARatioMode attribute), 52  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ARatioMode attribute), 34  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode attribute), 46  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ERatioMode attribute), 40  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode attribute), 59  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ERatioMode attribute), 53  
 B\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ARatioMode attribute), 64  
 B\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000aPiezoSine4000a resources.picotech.picoscope.enums.PS4000 attribute), 41  
 B\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000aPiezoSine5000a resources.picotech.picoscope.ps5000aPiezoSine5000a attribute), 92  
 B\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 50  
 B\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS6000 attribute), 64  
 Backend (class in msl.equipment.constants), 188  
 bandwidth (msl.equipment.resources.picotech.picoscope.channel.Parameters attribute), 13  
 BATCH\_AND\_SERIAL (msl.equipment.resources.picotech.picoscope.enums.PicoScopeAttribute), 14  
 BATCH\_AND\_SERIAL (msl.equipment.resources.picotech.picoscope.enums.PS2000GAttribute), 14  
 BATCH\_AND\_SERIAL (msl.equipment.resources.picotech.picoscope.enums.PS3000AAttribute), 16  
 BATCH\_AND\_SERIAL (msl.equipment.resources.picotech.picoscope.enums.PS3000ANDaSERIAL attribute), 24  
 BATCH\_AND\_SERIAL (msl.equipment.resources.picotech.picoscope.enums.PS4000AAttribute), 16  
 baud\_rate (msl.equipment.connection\_msl.ConnectionSerial attribute), 187  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS2000AAttribute), 13  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS2000GAttribute), 13  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS3000AAttribute), 17  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS3000ANDaSERIAL attribute), 24  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS4000AAttribute), 16  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS4000GAttribute), 17  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS5000AAttribute), 17  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS5000GAttribute), 17  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS6000AAttribute), 33  
 BELOW (msl.equipment.resources.picotech.picoscope.enums.PS6000ANDaSERIAL attribute), 33



- BW\_FULL (msl.equipment.resources.picotech.picoscope.enums.PS5000ABandwidthLimiter attribute), 54
- BW\_FULL (msl.equipment.resources.picotech.picoscope.enums.PS6000ABandwidthLimiter attribute), 59
- C**
- C (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel attribute), 18
- C (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel attribute), 14
- C (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel attribute), 28
- C (msl.equipment.resources.picotech.picoscope.enums.PS3000Channel attribute), 24
- C (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel attribute), 41
- C (msl.equipment.resources.picotech.picoscope.enums.PS4000Channel attribute), 35
- C (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 55
- C (msl.equipment.resources.picotech.picoscope.enums.PS5000Channel attribute), 50
- C (msl.equipment.resources.picotech.picoscope.enums.PS6000Channel attribute), 59
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex attribute), 18
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex attribute), 28
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex attribute), 41
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 34
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex attribute), 55
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000Channel attribute), 50
- C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 60
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex attribute), 18
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex attribute), 28
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex attribute), 41
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 34
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex attribute), 55
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 50
- C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 60
- Cage\_Rotator (msl.equipment.resources.thorlabs.kinesis.motion\_controller.enums.PS7000ABandwidthLimiter attribute), 54
- CAL\_DATE (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel attribute), 14
- CAL\_DATE (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel attribute), 14
- CAL\_DATE (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel attribute), 16
- CAL\_DATE (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 16
- calibration\_period (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel attribute), 193
- can\_home() (msl.equipment.resources.thorlabs.kinesis.integrated\_step\_controller.enums.PPC\_I attribute), 179
- can\_move\_without\_homing\_first() (msl.equipment.resources.thorlabs.kinesis.integrated\_step\_controller.enums.PPC\_I attribute), 178
- camera\_measurement() (msl.equipment.resources.bentham.benhw32.Bentham32Channel attribute), 10
- changePowerSource() (msl.equipment.resources.picotech.picoscope.picoscope\_a.enums.PS6000Channel attribute), 10
- changeToOddOrEven (msl.equipment.resources.thorlabs.kinesis.structs.NT\_TIAChannel attribute), 164
- CHANNELS (msl.equipment.resources.picotech.picoscope.channel.PicoChannel attribute), 13
- CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannel attribute), 47
- CHANNELS (msl.equipment.resources.picotech.picoscope.picoscope.PicoChannel attribute), 66
- CHANNELS (msl.equipment.resources.picotech.picoscope.structs.PS2000AChannel attribute), 96
- CHANNELS (msl.equipment.resources.picotech.picoscope.structs.PS2000Channel attribute), 96
- CHANNELS (msl.equipment.resources.picotech.picoscope.structs.PS2000Channel attribute), 96



attribute), 20 CHANNEL\_8 (msl.equipment.resources.picotech.picoscope.enums.  
CHANNEL\_24 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 29 CHANNEL\_8 (msl.equipment.resources.picotech.picoscope.enums.  
CHANNEL\_25 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 CHANNEL\_9 (msl.equipment.resources.picotech.picoscope.enums.  
CHANNEL\_25 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 CHANNEL\_9 (msl.equipment.resources.picotech.picoscope.enums.  
CHANNEL\_26 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 CHANNEL\_ANALOGUE\_OFFSET  
CHANNEL\_26 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 attribute), 47  
CHANNEL\_27 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 (msl.equipment.resources.picotech.picoscope.enums.PS4000ADigitalChannel  
CHANNEL\_27 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 CHANNEL\_COUPLING  
CHANNEL\_28 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 attribute), 47  
CHANNEL\_28 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 (msl.equipment.resources.picotech.picoscope.enums.PS4000ADigitalChannel  
CHANNEL\_29 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 CHANNEL\_MAX  
CHANNEL\_29 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 attribute), 20  
CHANNEL\_3 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 19 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
CHANNEL\_3 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 29 CHANNEL\_NAME  
CHANNEL\_30 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 attribute), 47  
CHANNEL\_30 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 (msl.equipment.resources.picotech.picoscope.enums.PS4000ADigitalChannel  
CHANNEL\_31 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 20 channelA (msl.equipment.resources.picotech.picoscope.structs.PS2000ADigitalChannel  
CHANNEL\_31 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 30 channelA (msl.equipment.resources.picotech.picoscope.structs.PS2000ADigitalChannel  
CHANNEL\_4 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 19 channelA (msl.equipment.resources.picotech.picoscope.structs.PS2000ADigitalChannel  
CHANNEL\_4 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 29 channelA (msl.equipment.resources.picotech.picoscope.structs.PS2000ADigitalChannel  
CHANNEL\_5 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 19 channelA (msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannel  
CHANNEL\_5 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 29 channelA (msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannel  
CHANNEL\_6 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 19 channelA (msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannel  
CHANNEL\_6 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 29 channelA (msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannel  
CHANNEL\_7 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel  
attribute), 19 channelA (msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannel  
CHANNEL\_7 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel  
attribute), 29 channelA (msl.equipment.resources.picotech.picoscope.structs.PS3000ADigitalChannel







attribute), 193			msl.equipment.resources.picotech.picoscope.helper),
ConnectionDemo (class in msl.equipment.connection_demo), 184		in 65	CURRENT_CLAMP_1000A (msl.equipment.resources.picotech.picoscope.enums.PS4000_CLAMP_1000A attribute), 36
ConnectionMessageBased (class in msl.equipment.connection_msl), 185		in 65	CURRENT_CLAMP_10A (msl.equipment.resources.picotech.picoscope.enums.PS4000_CLAMP_10A attribute), 36
ConnectionPyVISA (class in msl.equipment.connection_pyvisa), 187		in 65	CURRENT_MEASURING_DEVICE (msl.equipment.resources.picotech.picoscope.enums.PS4000_MEASURING_DEVICE attribute), 36
ConnectionRecord (class in msl.equipment.record_types), 193		in 65	current_power_source() (msl.equipment.resources.picotech.picoscope.picoscope_adapter.method), 72
connections() (msl.equipment.database.Database method), 190			CycleCount (msl.equipment.resources.thorlabs.kinesis.structs.KMOCycleCount attribute), 169
ConnectionSDK (class in msl.equipment.connection_msl), 184		in 65	cycleLength (msl.equipment.resources.thorlabs.kinesis.structs.PZ1CycleLength attribute), 165
ConnectionSerial (class in msl.equipment.connection_msl), 186		in 65	
constants() (msl.equipment.resources.utils.CHeader method), 179			
continuousCurrentLimit (msl.equipment.resources.thorlabs.kinesis.structs.KMOCycleLimit attribute), 162			
controlMode (msl.equipment.resources.thorlabs.kinesis.structs.KMOCycleMode attribute), 165			
controlSrc (msl.equipment.resources.thorlabs.kinesis.structs.KMOCycleSrc attribute), 166			
convert_message() (msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl static method), 156			
convert_to_enum() (msl.equipment.connection.Connection static method), 183			
countsPerUnit (msl.equipment.resources.thorlabs.kinesis.structs.KMOCycleCountsPerUnit attribute), 159			
coupling (msl.equipment.resources.picotech.picoscope.enums.PS5000_CH1_COUPLING attribute), 13			
CR (msl.equipment.connection_msl.ConnectionMessageBase attribute), 185			
create_callbacks_file() (in module msl.equipment.resources.picotech.picoscope.helper), 65			
create_picoscope_enums_file() (in module msl.equipment.resources.picotech.picoscope.helper), 65			
create_picoscope_functions_file() (in module msl.equipment.resources.picotech.picoscope.helper), 65			
create_picoscope_structs_file() (in module msl.equipment.resources.picotech.picoscope.helper), 65			
ctypes_map() (in module			

D

D (msl.equipment.resources.picotech.picoscope.enums.PS2000_ACQUISITION_MODE attribute), 18			
D (msl.equipment.resources.picotech.picoscope.enums.PS2000_CH1_COUPLING attribute), 14			
D (msl.equipment.resources.picotech.picoscope.enums.PS3000_ACQUISITION_MODE attribute), 28			
D (msl.equipment.resources.picotech.picoscope.enums.PS3000_CH1_COUPLING attribute), 24			
D (msl.equipment.resources.picotech.picoscope.enums.PS4000_ACQUISITION_MODE attribute), 41			
D (msl.equipment.resources.picotech.picoscope.enums.PS4000_CH1_COUPLING attribute), 35			
D (msl.equipment.resources.picotech.picoscope.enums.PS5000_ACQUISITION_MODE attribute), 55			
D (msl.equipment.resources.picotech.picoscope.enums.PS5000_CH1_COUPLING attribute), 50			
D (msl.equipment.resources.picotech.picoscope.enums.PS6000_ACQUISITION_MODE attribute), 60			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000_ACQUISITION_MODE attribute), 18			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000_ACQUISITION_MODE attribute), 28			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000_ACQUISITION_MODE attribute), 41			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000_CH1_COUPLING attribute), 34			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000_ACQUISITION_MODE attribute), 55			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000_CH1_COUPLING attribute), 50			
D_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000_ACQUISITION_MODE attribute), 60			

attribute), 60  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex attribute), 18  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex attribute), 28  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex attribute), 41  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 34  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex attribute), 55  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 50  
 D\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 60  
 data\_bits (msl.equipment.connection\_msl.ConnectionSerial method), 10  
 attribute), 187  
 Database (class in msl.equipment.database), 190  
 database() (msl.equipment.config.Config method), 182  
 DataBits (class in msl.equipment.constants), 189  
 date\_calibrated (msl.equipment.record\_types.EquipmentRecord attribute), 193  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS2000ACoupling attribute), 21  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS3000ACoupling attribute), 30  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS4000ACoupling attribute), 41  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS5000ACoupling attribute), 54  
 DC\_1M (msl.equipment.resources.picotech.picoscope.enums.PS4000Coupling attribute), 61  
 DC\_50R (msl.equipment.resources.picotech.picoscope.enums.PS6000Coupling attribute), 61  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 22  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS2000WaveType attribute), 17  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 31  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 43  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType attribute), 38  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 57  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType attribute), 52  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType attribute), 62  
 deadErrorBand (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_B attribute), 160  
 ChannelBufferIndex (msl.equipment.resources.thorlabs.kinesis.enums.PP attribute), 114  
 ChannelBufferIndex (msl.equipment.resources.thorlabs.kinesis.enums.PP attribute), 114  
 ChannelBufferIndex (msl.equipment.record\_types.EquipmentRecord attribute), 193  
 ChannelBufferIndex (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_L attribute), 157  
 DeviceCoupling (msl.equipment.resources.picotech.picoscope.enums.PS4000Coupling attribute), 61  
 device\_manager() (in module msl.equipment.resources.thorlabs.kinesis.motion\_control), 154  
 PS2000AWaveType (msl.equipment.resources.picotech.picoscope.ps4000a.Pic attribute), 44  
 PS3000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS4000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS4000WaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS5000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS6000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 diameter (msl.equipment.resources.thorlabs.kinesis.structs.NT\_Cir attribute), 44  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.M attribute), 44  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.M attribute), 44  
 delete\_group() (msl.equipment.resources.bentham.benhw32.Bentha attribute), 172  
 DEMO\_MODE (msl.equipment.config.Config attribute), 182  
 derivativeRecalculationTime (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_B attribute), 160  
 DerivativeRecalculationTime (msl.equipment.resources.thorlabs.kinesis.enums.PP attribute), 114  
 DerivativeRecalculationTime (msl.equipment.resources.thorlabs.kinesis.enums.PP attribute), 114  
 DerivativeRecalculationTime (msl.equipment.record\_types.EquipmentRecord attribute), 193  
 DerivativeRecalculationTime (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_L attribute), 157  
 DEVICE\_COUPLING (msl.equipment.resources.picotech.picoscope.enums.PS4000Coupling attribute), 61  
 device\_manager() (in module msl.equipment.resources.thorlabs.kinesis.motion\_control), 154  
 PS2000AWaveType (msl.equipment.resources.picotech.picoscope.ps4000a.Pic attribute), 44  
 PS3000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS4000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS4000WaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS5000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 PS6000AWaveType (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Ha attribute), 44  
 diameter (msl.equipment.resources.thorlabs.kinesis.structs.NT\_Cir attribute), 44  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.M attribute), 44  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.M attribute), 44

attribute), 162  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_LoopEncoderParams attribute), 167  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_LoopParameters attribute), 171  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_PIDParameters attribute), 171  
 differentialGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_PID\_LoopParameters attribute), 175  
 digIO1OperMode (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IO\_Settings method), 167  
 digIO1PulseWidth (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IO\_Settings attribute), 167  
 digIO1SignalMode (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IO\_Settings attribute), 167  
 digIO2OperMode (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IO\_Settings method), 167  
 digIO2PulseWidth (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IO\_Settings attribute), 167  
 digIO2SignalMode (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IO\_Settings attribute), 167  
 digital (msl.equipment.resources.picotech.picoscope.structs.PS2000A\_VIQ\_Conditions attribute), 96  
 digital (msl.equipment.resources.picotech.picoscope.structs.PS2000A\_Triggers\_Conditions attribute), 96  
 digital (msl.equipment.resources.picotech.picoscope.structs.PS3000A\_VIQ\_ConditionsV2 attribute), 99  
 digital (msl.equipment.resources.picotech.picoscope.structs.PS3000A\_Triggers\_ConditionsV2 attribute), 98  
 DIGITAL\_HARDWARE\_VERSION (msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfo attribute), 14  
 Dim (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_DisplayIntensity attribute), 116  
 direction (msl.equipment.resources.picotech.picoscope.structs.PS2000A\_DigitalChannelDirections attribute), 96  
 direction (msl.equipment.resources.picotech.picoscope.structs.PS3000A\_DigitalChannelDirections attribute), 99  
 direction (msl.equipment.resources.picotech.picoscope.structs.PS4000A\_Direction attribute), 101  
 direction (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_TripHomeingParameters attribute), 158  
 directionSense (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_JoystickParameters attribute), 160  
 DISABLE (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_TriggersWithinPreTrigger attribute), 58  
 disable\_channel() (msl.equipment.resources.thorlabs.kinesis.integrations.stepper\_motors.IntegratedStepperMotors method), 128  
 Disconnect (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LoopEncoderParams attribute), 167  
 disconnect() (msl.equipment.connection.Connection method), 100  
 disconnect() (msl.equipment.connection\_demo.ConnectionDemo method), 100  
 disconnect() (msl.equipment.connection\_msl.ConnectionSerial method), 100  
 disconnect() (msl.equipment.connection\_pyvisa.ConnectionPyVISA method), 100  
 Disconnect (msl.equipment.resources.bentham.benhw64.Bentham64\_FF\_IO\_Settings attribute), 11  
 disconnect() (msl.equipment.resources.picotech.picoscope.picoscope\_disconnect method), 11  
 disconnect() (msl.equipment.resources.thorlabs.fw102c.FilterWheel method), 11  
 disconnect() (msl.equipment.resources.thorlabs.kinesis.motion\_controller.MotionController method), 156  
 displayFFAdSettingsWindow() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10  
 displayFFAdSettingsWindow() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10  
 DisplayDimIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_Triggers\_Conditions attribute), 168  
 DisplayPS3000A\_VIQ\_ConditionsV2 (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MOT\_Triggers\_ConditionsV2 attribute), 170  
 DisplayDimIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KSC\_MOT\_Triggers\_Conditions attribute), 170  
 displayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MOT\_Triggers\_Conditions attribute), 170  
 DisplayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_Triggers\_Conditions attribute), 170  
 DisplayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KSC\_MOT\_Triggers\_Conditions attribute), 170  
 DisplayMode (msl.equipment.resources.thorlabs.kinesis.structs.TSC\_MOT\_TripHomeingParameters attribute), 170  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MOT\_TripHomeingParameters attribute), 170  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TripHomeingParameters attribute), 170  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KSC\_MOT\_TripHomeingParameters attribute), 170  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.TSC\_MOT\_TripHomeingParameters attribute), 170

DISTANCE (msl.equipment.resources.thorlabs.kinesis.integrattdistupper1motors.UnitType attribute), 127

DISTRIBUTION (msl.equipment.resources.picotech.picoscope.attributes.PS4000ARatioMode attribute), 46

DISTRIBUTION (msl.equipment.resources.picotech.picoscope.attributes.PS5000ARatioMode attribute), 59

DISTRIBUTION (msl.equipment.resources.picotech.picoscope.attributes.PS5000RatioMode attribute), 53

DISTRIBUTION (msl.equipment.resources.picotech.picoscope.attributes.PS6000RatioMode attribute), 64

DONT\_CARE (msl.equipment.resources.picotech.picoscope.attributes.PS2000ADigitalDirection attribute), 23

DONT\_CARE (msl.equipment.resources.picotech.picoscope.attributes.PS2000ATriggerState attribute), 24

DONT\_CARE (msl.equipment.resources.picotech.picoscope.attributes.PS2000TriggerState attribute), 17

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS3000ADigitalDirection attribute), 14

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS3000ATriggerState attribute), 33

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS3000TriggerState attribute), 27

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS3000ATriggerState attribute), 46

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS4000ATriggerState attribute), 39

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000ATriggerState attribute), 58

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000TriggerState attribute), 53

DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS6000ATriggerState attribute), 63

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS4000ASweepType attribute), 21

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000ASweepType attribute), 16

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000ASweepType attribute), 31

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS6000ASweepType attribute), 43

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS4000SweepType attribute), 37

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000ASweepType attribute), 56

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000SweepType attribute), 51

DOWN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS4000SweepType attribute), 61

downLimit (msl.equipment.resources.thorlabs.kinesis.structs.MSLEquipmentResource attribute), 164

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS2000ASweepType attribute), 20

E

EMAX (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS4000ASweepType attribute), 41

EMAX (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS5000SweepType attribute), 41

EMIN (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS4000SweepType attribute), 42

EFF\_10MHZ (msl.equipment.resources.picotech.picoscope.enums.PicoScope.PS2000ASweepType attribute), 59

attribute), 59 (msl.equipment.resources.picotech.picoscope.enums.PS3000ExternalFrequency)

EF\_25MHZ (msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency attribute), 59 ENTER\_OR\_EXIT

EF\_5MHZ (msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency attribute), 59 (msl.equipment.resources.picotech.picoscope.enums.PS3000ExternalFrequency attribute), 27

EF\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency attribute), 59 (msl.equipment.resources.picotech.picoscope.enums.PS4000ExternalFrequency attribute), 45

EF\_OFF (msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency attribute), 59 ENTER\_OR\_EXIT

EIGHT (msl.equipment.constants.DataBits attribute), 190 (msl.equipment.resources.picotech.picoscope.enums.PS4000ExternalFrequency attribute), 39

enable\_channel() (msl.equipment.resources.thorlabs.kinesis.integratedsteppermotors.integratedsteppermotors method), 128 (msl.equipment.resources.picotech.picoscope.enums.PS5000ExternalFrequency attribute), 58

enable\_last\_msg\_timer() (msl.equipment.resources.thorlabs.kinesis.integratedsteppermotors.integratedsteppermotors method), 126 (msl.equipment.resources.picotech.picoscope.enums.PS5000ExternalFrequency attribute), 53

enable\_last\_msg\_timer() (msl.equipment.resources.thorlabs.kinesis.integratedsteppermotors.integratedsteppermotors method), 128 (msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency attribute), 63

enable\_last\_msg\_timer() (msl.equipment.resources.thorlabs.kinesis.integratedsteppermotors.integratedsteppermotors method), 147 (msl.equipment.resources.picotech.picoscope.picoscope),

enabled (msl.equipment.resources.picotech.picoscope.channel.picoscopechannel attribute), 13 enums() (msl.equipment.resources.utils.CHeader)

encoding (msl.equipment.connection\_msl.ConnectionMessageBase attribute), 179 equipment (msl.equipment.database.Database attribute), 185

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS2000ThresholdDirection attribute), 23 equipment\_record

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdDirection attribute), 17 (msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdDirection attribute), 182

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdDirection attribute), 33 (msl.equipment.record\_types), 192

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection attribute), 27 (msl.equipment.resources.bentham.benhw64.Bentham method), 11

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection attribute), 45 (msl.equipment.resources.picotech.picoscope.picoscope.enums.PS4000ThresholdDirection attribute), 71

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection attribute), 39 (msl.equipment.resources.thorlabs.kinesis.motion\_controller.enums.PS4000ThresholdDirection attribute), 156

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdDirection attribute), 58 (msl.equipment.resources.thorlabs.fw102c.FilterWheel102c.enums.PS5000ThresholdDirection attribute), 176

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdDirection attribute), 53 (msl.equipment.resources.thorlabs.fw102c.FilterWheel102c.enums.PS6000ThresholdDirection attribute), 76

ENTER (msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdDirection attribute), 63 errcheck\_negative\_one()

ENTER\_OR\_EXIT (msl.equipment.resources.picotech.picoscope.picoscope\_2 (msl.equipment.resources.picotech.picoscope.enums.PS2000ThresholdDirection attribute), 23 errcheck\_non\_zero())

ENTER\_OR\_EXIT (msl.equipment.resources.thorlabs.fw102c.FilterWheel102c.enums.PS2000ThresholdDirection attribute), 17 errcheck\_one() (msl.equipment.resources.picotech.picoscope.picoscope.enums.PS2000ThresholdDirection attribute), 69

ENTER\_OR\_EXIT (msl.equipment.resources.thorlabs.fw102c.FilterWheel102c.enums.PS2000ThresholdDirection attribute), 69

errcheck\_true() (msl.equipment.resources.thorlabs.kinesis.structs.MQTTAttribute), 156

errcheck\_zero() (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 69

ERROR\_CODE (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 16

ERROR\_CODE (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 26

ETS (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 50

ETS\_CYCLE (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 50

ETS\_INTERLEAVE (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 50

ETS\_SAMPLE\_TIME\_PICOSECONDS (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 50

ETS\_STATE (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 50

EVEN (msl.equipment.constants.Parity attribute), 189

EVENT (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 34

excessEnergyLimit (msl.equipment.resources.thorlabs.kinesis.structs.MQTTAttribute), 162

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 23

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 17

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 33

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 27

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 45

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 39

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 58

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 53

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 63

EXT (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 18

EXT (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 14

EXT (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 28

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 24

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 41

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 35

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 55

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 50

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 60

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 50

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 32

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 45

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 38

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 57

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 52

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS6000 attribute), 63

EXIT (msl.equipment.resources.picotech.picoscope.ps2000a.PicoAttribute), 82

EXIT (msl.equipment.resources.picotech.picoscope.ps3000a.PicoAttribute), 85

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps3000a.PicoAttribute), 85

EXIT (msl.equipment.resources.picotech.picoscope.ps4000.PicoAttribute), 87

EXIT (msl.equipment.resources.picotech.picoscope.ps4000a.PicoAttribute), 87

EXIT (msl.equipment.resources.picotech.picoscope.ps4000a.PicoAttribute), 87

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoAttribute), 87

EXIT (msl.equipment.resources.picotech.picoscope.ps5000.PicoAttribute), 91

EXIT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoAttribute), 91

EXIT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoAttribute), 91

EXT\_MAX\_VOLTAGE (msl.equipment.resources.picotech.picoscope.ps5000a.PicoAttribute), 92

EXIT (msl.equipment.resources.picotech.picoscope.ps2000a.PicoAttribute), 82

EXT\_MIN\_VALUE (msl.equipment.resources.picotech.picoscope.ps2000a.PicoAttribute), 82





FALLING (msl.equipment.resources.picotech.picoscope.enums.PS6000.SigGenTrigType attribute), 62

FALLING (msl.equipment.resources.picotech.picoscope.enums.PS6000.ThresholdDirection attribute), 63

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS6000.ThresholdDirection attribute), 23

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS3600.ThresholdDirection attribute), 33

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 45

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 39

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 58

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS6000.ThresholdDirection attribute), 63

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS2000A.TriggerState attribute), 24

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS2000E.TriggerState attribute), 18

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS3000A.TriggerState attribute), 33

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS3000E.TriggerState attribute), 27

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS4000A.TriggerState attribute), 46

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS4000E.TriggerState attribute), 39

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS5000A.TriggerState attribute), 58

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS5000E.TriggerState attribute), 53

FALSE (msl.equipment.resources.picotech.picoscope.enums.PS6000E.TriggerState attribute), 63

FAST (msl.equipment.resources.picotech.picoscope.enums.PS2000A.EtsMode attribute), 21

FAST (msl.equipment.resources.picotech.picoscope.enums.PS2000E.EtsMode attribute), 16

FAST (msl.equipment.resources.picotech.picoscope.enums.PS3000A.EtsMode attribute), 31

FAST (msl.equipment.resources.picotech.picoscope.enums.PS3000E.EtsMode attribute), 27

FAST (msl.equipment.resources.picotech.picoscope.enums.PS4000A.EtsMode attribute), 43

FAST (msl.equipment.resources.picotech.picoscope.enums.PS4000E.EtsMode attribute), 43

FAST (msl.equipment.resources.picotech.picoscope.enums.PS6000.SigGenTrigType attribute), 62

FAST (msl.equipment.resources.picotech.picoscope.enums.PS5000.ThresholdDirection attribute), 61

FAST (msl.equipment.resources.thorlabs.fw102c.SpeedMode attribute), 75

FC\_20 (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 46

FC\_20 (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 40

FC\_200 (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 46

FC\_200 (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 40

FC\_20K (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 46

FC\_2K (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 46

FC\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 46

FC\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000.ThresholdDirection attribute), 46

feedbackSignal (msl.equipment.resources.thorlabs.kinesis.structs.BPCPC struct), 116

feedbackSrc (msl.equipment.resources.thorlabs.kinesis.structs.PPCPC struct), 116

feedForward (msl.equipment.resources.thorlabs.kinesis.structs.MO struct), 116

FF\_InputButton (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOModes), 116

FF\_InputLogic (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOModes), 116

FF\_InputSwap (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOModes), 116

FF\_OutputHighAtSetPosition (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOModes), 116

FF\_OutputLowAtSetPosition (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOModes), 116

FF\_OutputMoving (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOModes), 116

(msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOMode, 14  
attribute), 116 FIRMWARE\_VERSION\_2  
FF\_OutputLevel (msl.equipment.resources.thorlabs.kinesis.enums.FF\_SignalModes (msl.equipment.resources.picotech.picoscope.enums.Pico  
attribute), 116 attribute), 14  
FF\_OutputPulse (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOMode (SignalModes) (msl.equipment.resources.thorlabs.kinesis.structs.F  
attribute), 116 attribute), 158  
FF\_OutputSwap (msl.equipment.resources.thorlabs.kinesis.enums.FIRST\_USB (FF\_SignalModes) (msl.equipment.resources.picotech.picoscope.ps2000.  
attribute), 116 attribute), 81  
FF\_PositionError (msl.equipment.resources.thorlabs.kinesis.enums.FIRST\_USB (FF\_Position) (msl.equipment.resources.picotech.picoscope.ps3000.  
attribute), 116 attribute), 83  
FF\_Positions (class in FIVE (msl.equipment.constants.DataBits at-  
msl.equipment.resources.thorlabs.kinesis.enums), attribute), 189  
116 flash\_led() (msl.equipment.resources.picotech.picoscope.picoscope  
FF\_SetPositionOnPositiveEdge method), 70  
(msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOMode (m  
attribute), 116 method), 72  
FF\_SignalModes (class in forceCalibration (msl.equipment.resources.thorlabs.kinesis.structs.F  
msl.equipment.resources.thorlabs.kinesis.enums), attribute), 174  
116 Forward (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_Di  
FF\_ToggleOnPositiveEdge attribute), 121  
(msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOMode (m  
attribute), 116 attribute), 166  
filter1Fc (msl.equipment.resources.thorlabs.kinesis.structs.FREQUENCY\_COUNTER  
attribute), 166 (msl.equipment.resources.picotech.picoscope.enums.PS40  
filter1Q (msl.equipment.resources.thorlabs.kinesis.structs.PPCatNotchParams  
attribute), 166 FREQUENCY\_COUNTER\_CHANNEL  
filter2Fc (msl.equipment.resources.thorlabs.kinesis.structs.PPCatNotchParams (m  
attribute), 166 attribute), 48  
filter2Q (msl.equipment.resources.thorlabs.kinesis.structs.FREQUENCY\_COUNTER\_ENABLED  
attribute), 166 (msl.equipment.resources.picotech.picoscope.enums.PS40  
Filter\_Flipper (msl.equipment.resources.thorlabs.kinesis.motioncontrol.MotionControl  
attribute), 154 FREQUENCY\_COUNTER\_RANGE  
Filter\_Wheel (msl.equipment.resources.thorlabs.kinesis.motioncontrol.MotionControl (m  
attribute), 154 attribute), 48  
FilterCount (class in FREQUENCY\_COUNTER\_TRESHOLDMAJOR  
msl.equipment.resources.thorlabs.fw102c), (msl.equipment.resources.picotech.picoscope.enums.PS40  
175 attribute), 48  
FilterFlipper (class in FREQUENCY\_COUNTER\_TRESHOLDMINOR  
msl.equipment.resources.thorlabs.kinesis.filter\_flipper), (msl.equipment.resources.picotech.picoscope.enums.PS40  
123 attribute), 48  
filterNo (msl.equipment.resources.thorlabs.kinesis.structs.FS (RPCatNotchParams) (m  
attribute), 166 attribute), 21  
FilterWheel102C (class in FS (msl.equipment.resources.picotech.picoscope.enums.PS2000Tir  
msl.equipment.resources.thorlabs.fw102c), attribute), 15  
175 FS (msl.equipment.resources.picotech.picoscope.enums.PS3000AT  
find\_sdk\_class() (in module attribute), 31  
msl.equipment.resources), 8 FS (msl.equipment.resources.picotech.picoscope.enums.PS3000Tir  
find\_serial\_class() (in module attribute), 25  
msl.equipment.resources), 8 FS (msl.equipment.resources.picotech.picoscope.enums.PS4000AT  
FIRMWARE\_VERSION\_1 attribute), 43  
(msl.equipment.resources.picotech.picoscope.enums.FS (msl.equipment.resources.picotech.picoscope.enums.PS4000Tir

attribute), 37

FS (msl.equipment.resources.picotech.picoscope.enums.PS5000AAttribute), 165

attribute), 56

FS (msl.equipment.resources.picotech.picoscope.enums.PS5000FilterUnit), 73

attribute), 51

FS (msl.equipment.resources.picotech.picoscope.enums.PS6000FilterUnit), 73

attribute), 61

FT\_DeviceNotFound (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_HIGH), 44

attribute), 107

FT\_DeviceNotOpened (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_HIGH), 57

attribute), 107

FT\_DeviceNotPresent (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_HIGH), 52

attribute), 107

FT\_IncorrectDevice (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_HIGH), 22

attribute), 107

FT\_InsufficientResources (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_LOW), 32

attribute), 107

FT\_InvalidHandle (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_LOW), 38

attribute), 107

FT\_InvalidParameter (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_LOW), 57

attribute), 107

FT\_IOError (msl.equipment.resources.thorlabs.kinesis.enums.GATE\_LOW), 62

attribute), 107

FT\_OK (msl.equipment.resources.thorlabs.kinesis.enums.GAUSSIAN), 22

attribute), 107

FT\_Status (class in GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.F), 17

msl.equipment.resources.thorlabs.kinesis.enums), attribute), 17

107

functions() (msl.equipment.resources.utils.CHeader attribute), 31

method), 179

futureUse (msl.equipment.resources.thorlabs.kinesis.structs.TSGSettings), 174

attribute), 174

FW\_FAIL (msl.equipment.resources.picotech.picoscope.enums.PS2000Error), 15

attribute), 15

FW\_FAIL (msl.equipment.resources.picotech.picoscope.enums.PS3000Error), 57

attribute), 26

attribute), 52

G GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.F), 2

G (msl.equipment.resources.picotech.picoscope.enums.PS4000Attribute), 41

attribute), 41

G\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex), 83

attribute), 42

G\_MIN (msl.equipment.resources.picotech.picoscope.enums.GAUSSIAN\_MAX\_FREQUENCY), 42

attribute), 42

(msl.equipment.resources.picotech.picoscope.ps3000a.Pic

attribute), 86

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000a attribute), 90

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000a attribute), 91

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000a attribute), 92

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope6000a attribute), 94

GD\_Trig\_High (msl.equipment.resources.thorlabs.kinesis.enums.GD\_TriggerPolarities attribute), 120

GD\_Trig\_Low (msl.equipment.resources.thorlabs.kinesis.enums.GD\_TriggerPolarities attribute), 120

get() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 9

get() (msl.equipment.resources.bentham.benhw64.Bentham64 method), 11

get\_acceleration() (msl.equipment.resources.thorlabs.fw102c.FilterFlipper102C method), 176

get\_analogue\_offset() (msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScopeApi method), 72

get\_backlash() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 128

get\_bow\_index() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 129

get\_button\_params() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 129

get\_button\_params\_block() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 129

get\_c\_group() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 9

get\_calibration\_file() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 129

get\_channel\_information() (msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScopeApi method), 72

get\_class() (in module msl.equipment.resources), 8

get\_common\_mode\_overflow() (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000a method), 90

get\_component\_list() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 9

get\_component\_list() (msl.equipment.resources.bentham.benhw64.Bentham64 method), 11

get\_cycle\_params() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid method), 147

get\_cycle\_params\_block() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid method), 147

get\_device\_info() (msl.equipment.resources.thorlabs.kinesis.motion\_control.QuKPA method), 1056

get\_device\_list() (msl.equipment.resources.thorlabs.kinesis.motion\_control.QuKPA method), 1056

get\_device\_list\_size() (msl.equipment.resources.thorlabs.kinesis.motion\_control.QuKPA static method), 156

get\_resolution() (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000a method), 92

get\_device\_unit\_from\_real\_value() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 129

get\_digital\_outputs() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid method), 147

get\_firmware\_version() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 129

get\_group() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 9

get\_group() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper102C method), 124

get\_hardware\_info() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 130

get\_hardware\_info() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid method), 147

get\_hardware\_info\_block() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 130

get\_hardware\_info\_block() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid method), 147

get\_hardware\_type() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 9

method), 9

get\_hardware\_type() (msl.equipment.resources.bentham.benhw64.Bentham64 method), 11

get\_homing\_params\_block() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 130

get\_homing\_velocity() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 130

get\_hub\_bay() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid module), 147

get\_id() (msl.equipment.resources.thorlabs.fw102c.FilterWheel102c method), 176

get\_io\_settings() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 125

get\_jog\_mode() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 130

get\_jog\_params\_block() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 130

get\_jog\_step\_size() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 130

get\_jog\_vel\_params() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 130

get\_led\_switches() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 131

get\_led\_switches() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid module), 148

get\_limit\_switch\_params() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 131

get\_limit\_switch\_params\_block() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors method), 131

get\_lines() (in module msl.equipment.resources.utils), 178

get\_lines() (msl.equipment.resources.utils.CHeader method), 179

get\_log() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10

get\_log\_size() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10

get\_max\_bw() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10

get\_max\_down\_sample\_ratio() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi method), 9

get\_max\_ets\_values() (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScopeApi method), 86

get\_max\_segments() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 72

get\_max\_velocity() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 72

get\_mmi\_params() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 148

get\_mmi\_params\_ext() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 148

get\_mono\_items() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 9

get\_motor\_params() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 131

get\_motor\_params\_ext() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 132

get\_motor\_travel\_limits() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 132

get\_motor\_travel\_mode() (mstequipmentresourcesintegratedsteppermotorsintegratedsteppermotors method), 132

get\_motor\_velocity\_limits() (msl.equipment.resources.thorlabs.kinesis.integrated\_step method), 132

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.integrated\_step method), 132

get\_move\_distance() (msl.equipment.resources.thorlabs.kinesis.integrated\_step method), 132

get\_n\_groups() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10



(msl.equipment.resources.picotech.picoscope.picoscope\_2k3k.PicoScope2k3k  
 method), 70  
 get\_string() (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000A  
 method), 90  
 get\_struct\_imports() (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k.PicoScope2k3k  
 method), 66  
 (msl.equipment.resources.utils.CHeader method), 180  
 get\_text\_between\_brackets() (msl.equipment.resources.utils.CHeader static method), 180  
 get\_time\_to\_current\_pos() (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C), 74  
 method), 177  
 get\_timebase() (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k.PicoScope2k3k  
 method), 70  
 get\_timebase() (msl.equipment.resources.picotech.picoscope.picoscope\_picoapi.PicoScopeApi  
 method), 73  
 get\_timebase2() (msl.equipment.resources.picotech.picoscope.picoscope\_picoapi.PicoScopeApi  
 method), 73  
 get\_times\_and\_values() (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k.PicoScope2k3k  
 method), 70  
 get\_transit\_time() (msl.equipment.resources.thorlabs.kinesis.filterflipper.FilterFlipper  
 method), 125  
 get\_trigger\_channel\_time\_offset() (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000  
 method), 88  
 get\_trigger\_channel\_time\_offset64() (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000  
 method), 88  
 get\_trigger\_config\_params() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid  
 method), 149  
 get\_trigger\_config\_params\_block() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid  
 method), 149  
 get\_trigger\_info\_bulk() (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000A  
 method), 86  
 get\_trigger\_info\_bulk() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepmotors.IntegratedStepperMotors  
 method), 135  
 (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000A), 10  
 get\_trigger\_mode() (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C), 9  
 method), 177  
 get\_trigger\_switches() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepmotors.IntegratedStepperMotors  
 method), 135  
 get\_trigger\_time\_offset() (msl.equipment.resources.picotech.picoscope.picoscope\_picoapi.PicoScopeApi  
 method), 73



(msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.equipment.resources.picotech.picoscope.enums.  
attribute), 18  
attribute), 62

GREATER\_THAN HALF\_SINE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.enums.PS3000A\_Pulse\_Width\_Type picotech.picoscope.ps2000a.Pico  
attribute), 34  
attribute), 83

GREATER\_THAN HALF\_SINE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.enums.PS3000A\_Pulse\_Width\_Type picotech.picoscope.ps3000a.Pico  
attribute), 28  
attribute), 86

GREATER\_THAN HALF\_SINE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.enums.PS4000A\_Pulse\_Width\_Type picotech.picoscope.ps4000a.Pico  
attribute), 47  
attribute), 90

GREATER\_THAN HALF\_SINE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.enums.PS4000A\_Pulse\_Width\_Type picotech.picoscope.ps5000.Pico  
attribute), 40  
attribute), 91

GREATER\_THAN HALF\_SINE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.enums.PS5000A\_Pulse\_Width\_Type picotech.picoscope.ps5000a.Pico  
attribute), 59  
attribute), 92

GREATER\_THAN HALF\_SINE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.enums.PS5000A\_Pulse\_Width\_Type picotech.picoscope.ps6000.Pico  
attribute), 54  
attribute), 94

GREATER\_THAN handle (msl.equipment.resources.picotech.picoscope.picoscope.Pico  
(msl.equipment.resources.picotech.picoscope.enums.PS6000A\_Pulse\_Width\_Type  
attribute), 64  
attribute), 16

GREEN (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_Channelsources.picotech.picoscope.enums.Pico  
attribute), 44  
attribute), 14

group\_add() (msl.equipment.resources.bentham.benhw32.Berithan)325  
method), 9  
(msl.equipment.resources.picotech.picoscope.enums.PS2000A)325

group\_remove() (msl.equipment.resources.bentham.benhw32.Berithan)325  
method), 9  
HARDWARE\_VERSION  
(msl.equipment.resources.picotech.picoscope.enums.PS3000A)26

**H**  
attribute), 26

H (msl.equipment.resources.picotech.picoscope.enums.hs.PS4000A\_Channelsources.picotech.picoscope.enums.Pico  
attribute), 41  
attribute), 158

H\_MAX (msl.equipment.resources.picotech.picoscope.enums.hs.PS4000A\_ChannelBufferIndex  
attribute), 42  
(msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper)16

H\_MIN (msl.equipment.resources.picotech.picoscope.enums.hs.PS4000A\_ChannelBufferIndex  
attribute), 42  
has\_last\_msg\_timer\_overrun()

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType.thorlabs.kinesis.integrated\_step  
attribute), 22  
method), 135

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType  
attribute), 17  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid)16

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType  
attribute), 31  
HIGH (msl.equipment.resources.picotech.picoscope.enums.PS2000A)325

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType  
attribute), 43  
HIGH (msl.equipment.resources.picotech.picoscope.enums.PS3000A)325

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType  
attribute), 38  
highGearAcceleration

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType.thorlabs.kinesis.structs.MOT\_J  
attribute), 57  
attribute), 160

HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.HALF\_SINE\_PULSE\_WIDTH\_TYPE (msl.EquipmentWaveType  
attribute), 52  
(msl.equipment.resources.thorlabs.kinesis.structs.MOT\_J

attribute), 160

hold\_off() (msl.equipment.resources.picotech.picoscope.picoscope.PicoScopeApi method), 75

home() (msl.equipment.resources.thorlabs.kinesis.filter\_flipped.FlippedFlipper method), 124

home() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers.IntegratedStepperMotors method), 135

horizontalComponent (msl.equipment.resources.thorlabs.kinesis.stubs.HMCComponents attribute), 163

hubAnalogOutput (msl.equipment.resources.thorlabs.kinesis.stubs.TSGM10Sensors attribute), 174

HubAnalogueModes (class in msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerChannelProperties msl.equipment.resources.thorlabs.kinesis.enums), attribute, 33

119

HV (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IOOutputMode attribute), 115

hysteresis (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 94

hysteresis (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 97

hysteresis (msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerChannelProperties attribute), 103

hysteresisLower (msl.equipment.resources.picotech.picoscope.structs.PS6000TriggerChannelProperties attribute), 106

hysteresisUpper (msl.equipment.resources.picotech.picoscope.structs.PS6000TriggerChannelProperties attribute), 106

I

identify() (msl.equipment.resources.thorlabs.kinesis.filter\_flipped.FlippedFlipper method), 124

identify() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers.IntegratedStepperMotors method), 135

identify() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoids.KCubeSolenoid method), 149

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS2000APulseWidthType attribute), 24

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS2000PulseWidthType attribute), 18

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS3000APulseWidthType attribute), 34

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS3000PulseWidthType attribute), 28

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS4000APulseWidthType attribute), 47

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS4000PulseWidthType attribute), 40

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS5000APulseWidthType attribute), 59

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS5000PulseWidthType attribute), 54

IN\_RANGE (msl.equipment.resources.picotech.picoscope.enums.PS2000PulseWidthType attribute), 23

IN\_RANGE (msl.equipment.resources.thorlabs.kinesis.stubs.TSGM10Sensors attribute), 17

INSIDE (msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerChannelProperties attribute), 33

INSIDE (msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerChannelProperties attribute), 33

INSIDE (msl.equipment.resources.picotech.picoscope.enums.PS4000TriggerChannelProperties attribute), 33

INSIDE (msl.equipment.resources.picotech.picoscope.enums.PS5000TriggerChannelProperties attribute), 33

INSIDE (msl.equipment.resources.picotech.picoscope.enums.PS6000TriggerChannelProperties attribute), 33

INSIDE (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralLimit (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralLimit (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralLimit (msl.equipment.resources.thorlabs.kinesis.structs.MOTOR\_PROPERTIES attribute), 161

integralTerm (msl.equipment.resources.thorlabs.kinesis.structs.PZ\_PROPERTIES attribute), 161

IntegratedStepperMotors (class in msl.equipment.resources.thorlabs.kinesis.integrated\_steppers)

interface (msl.equipment.record\_types.ConnectionRecord attribute), 168  
 attribute), 194  
 JoystickMode (msl.equipment.resources.thorlabs.kinesis.structs.KP attribute), 170  
 is\_calibration\_active()  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 135  
**K**  
 is\_led\_flashing()  
 (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi method), 75  
 is\_open()  
 (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C attribute), 154  
 method), 177  
 KCube\_DC\_Servo  
 is\_ready()  
 (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScope method), 67  
 attribute), 154  
 is\_trigger\_or\_pulse\_width\_qualifier\_enabled()  
 KCube\_LaserSource  
 (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi method), 75  
 attribute), 154  
 isCustomType (msl.equipment.resources.thorlabs.kinesis.structs.KCubePic7D attribute), 157  
 (msl.equipment.resources.thorlabs.kinesis.motion\_control.resources.thorlabs.kinesis.motion\_control attribute), 154  
 isKnownType (msl.equipment.resources.thorlabs.kinesis.structs.KCubePic7D attribute), 157  
 (msl.equipment.resources.thorlabs.kinesis.motion\_control.resources.thorlabs.kinesis.motion\_control attribute), 155  
 isLaser (msl.equipment.resources.thorlabs.kinesis.structs.KCubePic7D attribute), 157  
 (msl.equipment.resources.thorlabs.kinesis.motion\_control.resources.thorlabs.kinesis.motion\_control attribute), 155  
 isPiezoDevice (msl.equipment.resources.thorlabs.kinesis.structs.KCubePic7D attribute), 157  
 KCubeSolenoid (class in  
 msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) 146  
 isRack (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Developer attribute), 157  
 KD\_TrigOut\_Diff  
 (msl.equipment.resources.thorlabs.kinesis.enums.QD\_KP attribute), 120  
**J**  
 jerk (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_VelocityProfileParameters attribute), 159  
 KD\_TrigOut\_GPO  
 Jog (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_ButtselMode attribute), 121  
 (msl.equipment.resources.thorlabs.kinesis.enums.QD\_KP attribute), 120  
 JogContinuous (msl.equipment.resources.thorlabs.kinesis.enums.KD\_TrigOut\_BV attribute), 121  
 (msl.equipment.resources.thorlabs.kinesis.enums.QD\_KP attribute), 120  
 JogStep (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_AttrMode attribute), 121  
 KD\_TrigOut\_SumDiff  
 Joystick (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IO\_EquipmentMode attribute), 115  
 (msl.equipment.resources.thorlabs.kinesis.enums.QD\_KP attribute), 120  
 JoystickAcceleration  
 KERNEL\_DRIVER\_VERSION  
 (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_MMIPParams attribute), 168  
 (msl.equipment.resources.picotech.picoscope.enums.PS30 attribute), 16  
 JoystickBnc (msl.equipment.resources.thorlabs.kinesis.structs.KERNEL\_PPC\_IO\_EquipmentMode attribute), 115  
 (msl.equipment.resources.picotech.picoscope.enums.PS30 attribute), 26  
 JoystickDirectionSense  
 (msl.equipment.resources.thorlabs.kinesis.structs.KERNEL\_MMIPParams attribute), 168  
 (msl.equipment.resources.picotech.picoscope.enums.Pico attribute), 14  
 JoystickDirectionSense  
 (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_Cozs\_MMIPParams attribute), 170  
 (msl.equipment.resources.thorlabs.kinesis.enums.KLS\_Or attribute), 118  
 JoystickMax Velocity  
 (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_KMOT\_MMIPParams attribute), 168  
 (msl.equipment.resources.thorlabs.kinesis.enums.KLS\_Or attribute), 118  
 JoystickMode (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_MMIPParams attribute), 168

KLS\_Disabled (msl.equipment.resources.thorlabs.kinesis.enums.KLS\_TriggerMode attribute), 118  
 KLS\_HighStability (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_JS\_Jog (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_JS\_Positive (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_JS\_Velocity (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 117  
 KLS\_Input (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_JS\_Negative (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_JS\_Positive (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_JS\_Velocity (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_InterlockEnabled (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_LaserOn (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 116  
 KLS\_LowStability (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 116  
 KLS\_MMIPParams (class in msl.equipment.resources.thorlabs.kinesis.structs), 169  
 KLS\_ModulationTrigger (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_OpMode (class in msl.equipment.resources.thorlabs.kinesis.enums), 117  
 KLS\_Output (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_SetPointChange (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_SetPower (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_TriggerMode (class in msl.equipment.resources.thorlabs.kinesis.enums), 118  
 KLS\_TrigIOPParams (class in msl.equipment.resources.thorlabs.kinesis.structs), 169  
 KLS\_TrigPol\_High (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_TrigPol\_Low (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerMode attribute), 118  
 KLS\_TrigPolarity (class in msl.equipment.resources.thorlabs.kinesis.enums), 118  
 KMOT\_JoystickDirectionSense (class in msl.equipment.resources.thorlabs.kinesis.enums), 116  
 KMOT\_JoyStickMode (class in msl.equipment.resources.thorlabs.kinesis.enums), 117

KMOT\_TrigOut\_GPO (msl.equipment.resources.thorlabs.kinesis.structs),  
 (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerPortMode  
 attribute), 117 KPZ\_TriggerPortMode (class in  
 KMOT\_TrigOut\_InMotion (msl.equipment.resources.thorlabs.kinesis.enums),  
 (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerPortMode  
 attribute), 117 KPZ\_TriggerPortPolarity (class in  
 KMOT\_TrigOut\_Synch (msl.equipment.resources.thorlabs.kinesis.enums),  
 (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerPortMode  
 attribute), 117 KPZ\_TrigIn\_GPI (msl.equipment.resources.thorlabs.kinesis.enums  
 attribute), 119  
 KMOT\_TrigPolarityHigh (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_VoltageSensePolarity  
 attribute), 117 (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_Tr  
 KMOT\_TrigPolarityLow (msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_VoltageSensePolarity  
 attribute), 117 (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_Tr  
 KPZ\_JoyStickChangeRate (class in msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_TrigOut\_GPO  
 118 (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_Tr  
 KPZ\_JoystickDirectionSense (class in msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_TrigPolarityHigh  
 118 (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_Tr  
 KPZ\_JoyStickMode (class in msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_TrigPolarityLow  
 118 (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_Tr  
 KPZ\_JS\_High (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_JoyStickChangeRate  
 attribute), 118 KSC\_MMIPParams (class in  
 KPZ\_JS\_JogVoltage (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_JoyStickMode  
 attribute), 118 KSC\_TrigDisabled  
 KPZ\_JS\_Low (msl.equipment.resources.thorlabs.kinesis.enums.KPZ\_JoyStickChangeRate  
 attribute), 118 (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_Tr  
 KPZ\_JS\_Medium (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_FrequencyCutoff\_KPZ\_JoyStickChangeRate  
 attribute), 118 in  
 KPZ\_JS\_MoveAtVoltage (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_KPZ\_JoyStickMode  
 attribute), 118 msl.equipment.resources.thorlabs.kinesis.structs),  
 174  
 KPZ\_JS\_Negative (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_KPZ\_JoyStickMode  
 attribute), 118 (msl.equipment.resources.thorlabs.kinesis.enums),  
 KPZ\_JS\_Positive (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_KPZ\_JoyStickMode  
 attribute), 118 KSC\_KPZ\_JoyStickPolaritySense (class in  
 KPZ\_JS\_SetVoltage (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_TrigIn\_GPI  
 attribute), 121 (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_Tr  
 KPZ\_MMIPParams (class in msl.equipment.resources.thorlabs.kinesis.structs.KSC\_TrigPolarityHigh  
 170 (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_Tr  
 KPZ\_TrigDisabled (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_KPZ\_TriggerPortMode  
 attribute), 119 (msl.equipment.resources.thorlabs.kinesis.enums.KSC\_Tr  
 KPZ\_TriggerConfig (class in msl.equipment.resources.thorlabs.kinesis.structs.KSC\_TrigPolarityHigh  
 attribute), 121

L

- LD\_SoftwareOnly (msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags attribute), 122
- LabJack\_050 (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 155
- LabJack\_490 (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 155
- last\_button\_press() (msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000 method), 81
- LAST\_USB (msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000 attribute), 81
- LAST\_USB (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000 attribute), 83
- lastNotUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessCurrentLoopParameters attribute), 161
- lastNotUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessElectricOutputParameters attribute), 162
- lastNotUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessPositionLoopParameters attribute), 160
- lastNotUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessTrackSettleParameters attribute), 161
- lastNotUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_VelocityProfileParameters attribute), 159
- LD\_AnodeGrounded (msl.equipment.resources.thorlabs.kinesis.enums.LD\_POLARITY attribute), 122
- LD\_CathodeGrounded (msl.equipment.resources.thorlabs.kinesis.enums.LD\_POLARITY attribute), 122
- LD\_DisplayUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), 122
- LD\_ExternalSignal (msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags attribute), 122
- LD\_ILD (msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits attribute), 122
- LD\_ILim (msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits attribute), 122
- LD\_InputSourceFlags (class in msl.equipment.resources.thorlabs.kinesis.enums), 122
- LD\_IPD (msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits attribute), 122
- LD\_PLD (msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits attribute), 122
- LD\_POLARITY (class in msl.equipment.resources.thorlabs.kinesis.enums), 122
- LD\_Potentiometer (msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags attribute), 122
- LD\_TIA\_100uA (msl.equipment.resources.thorlabs.kinesis.enums.LD\_TIA attribute), 122
- LD\_TIA\_10mA (msl.equipment.resources.thorlabs.kinesis.enums.LD\_TIA attribute), 122
- LD\_TIA\_1mA (msl.equipment.resources.thorlabs.kinesis.enums.LD\_TIA attribute), 122
- LD\_TIA\_RANGES (class in msl.equipment.resources.thorlabs.kinesis.enums), 122
- LESS\_THAN (msl.equipment.resources.picotech.picoscope.enums.LESS\_THAN attribute), 28
- LESS\_THAN (msl.equipment.resources.picotech.picoscope.enums.LESS\_THAN attribute), 40
- LESS\_THAN (msl.equipment.resources.picotech.picoscope.enums.LESS\_THAN attribute), 59
- LESS\_THAN (msl.equipment.resources.picotech.picoscope.enums.LESS\_THAN attribute), 54
- LESS\_THAN (msl.equipment.resources.picotech.picoscope.enums.LESS\_THAN attribute), 64
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS2000 attribute), 17
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS3000 attribute), 32
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS3000 attribute), 27
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS4000 attribute), 45
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS4000 attribute), 39
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 57
- LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 53

LEVEL (msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdMode attribute), 63

LF (msl.equipment.connection\_msl.ConnectionMessageBaseAcceleration attribute), 185

limitSwitch (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_JointParameters attribute), 158

load\_settings() (msl.equipment.resources.thorlabs.kinesis.filter.flipEquipmentFlippers resources.thorlabs.kinesis.structs.MOT\_JointParameters attribute), 124

load\_settings() (msl.equipment.resources.thorlabs.kinesis.kinow.PastFilterCutOffFreq motors.IntegratedStepperMotors attribute), 135

load\_settings() (msl.equipment.resources.thorlabs.kinesis.kcube.kcube.KCubeSolenoid attribute), 149

load\_setup() (msl.equipment.resources.bentham.benhw32.BenthamEquipment.resources.thorlabs.kinesis.structs.QD\_LowPassFilterCutOffFreq attribute), 9

load\_setup() (msl.equipment.resources.bentham.benhw32.BenthamEquipment.resources.thorlabs.kinesis.structs.QD\_LowPassFilterEnabled attribute), 11

location (msl.equipment.record\_types.EquipmentRecord attribute), 193

log\_critical() (msl.equipment.connection.Connection attribute), 183

log\_debug() (msl.equipment.connection.Connection lowVoltageOutputRoute attribute), 183

log\_errcheck() (msl.equipment.connection\_msl.ConnectionSDAttribute), 185

log\_error() (msl.equipment.connection.Connection attribute), 183

log\_info() (msl.equipment.connection.Connection lowVoltageOutRange attribute), 183

log\_warning() (msl.equipment.connection.Connection attribute), 183

LONG\_PRESS (msl.equipment.resources.picotech.picoscope.enums.PS2000CustomStart attribute), 16

Long\_Travel\_Stage (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl resources.thorlabs.kinesis.enums.LS\_Input attribute), 155

loopMode (msl.equipment.resources.thorlabs.kinesis.LS\_InputMSOTireEncoderParams in attribute), 167

LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000 attribute), 81

LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000 attribute), 84

LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000 attribute), 87

LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A attribute), 89

LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps5000.PicoScope5000 attribute), 91

LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000A attribute), 91

LOW (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection attribute), 23





MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType  
attribute), 38

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ChannelOffsetLimit  
attribute), 55

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AEMode  
attribute), 56

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AnalogMode  
attribute), 57

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000SweepOffset  
attribute), 56

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AnalogUnits  
attribute), 56

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000TriggerState  
attribute), 58

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType  
attribute), 57

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex  
attribute), 50

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000EquiMode  
attribute), 51

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelOffset  
attribute), 52

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000StepType  
attribute), 51

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000Impedance  
attribute), 51

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelOffset  
attribute), 53

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType  
attribute), 52

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex  
attribute), 60

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000CGMMode  
attribute), 61

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000IndexMode  
attribute), 62

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000SupplyType  
attribute), 61

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelOffset  
attribute), 61

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000TriggerState  
attribute), 63

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000WaveType  
attribute), 62

MAX\_4\_CHANNELS  
(msl.equipment.resources.picotech.picoscope.enums.PS4000Channels  
attribute), 41

MAX\_ACCELEROMETER  
(msl.equipment.resources.picotech.picoscope.enums.PS4000Range  
attribute), 36

MAX\_ANALOGUE\_OFFSET\_500MV\_2V

MAX\_ANALOGUE\_OFFSET\_500MV\_2V  
(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
attribute), 82

MAX\_ANALOGUE\_OFFSET\_500MV\_2V  
(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
attribute), 82

MAX\_ANALOGUE\_OFFSET\_500MV\_2V  
(msl.equipment.resources.picotech.picoscope.ps4000a.Pico  
attribute), 89

MAX\_ANALOGUE\_OFFSET\_500MV\_2V  
(msl.equipment.resources.picotech.picoscope.ps5000a.Pico  
attribute), 93

MAX\_ANALOGUE\_OFFSET\_500MV\_2V  
(msl.equipment.resources.picotech.picoscope.ps6000.Pico  
attribute), 93

MAX\_ANALOGUE\_OFFSET\_50MV\_200MV  
(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
attribute), 82

MAX\_ANALOGUE\_OFFSET\_50MV\_200MV  
(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
attribute), 85

MAX\_ANALOGUE\_OFFSET\_50MV\_200MV  
(msl.equipment.resources.picotech.picoscope.ps4000a.Pico  
attribute), 89

MAX\_ANALOGUE\_OFFSET\_50MV\_200MV  
(msl.equipment.resources.picotech.picoscope.ps5000a.Pico  
attribute), 92

MAX\_ANALOGUE\_OFFSET\_50MV\_200MV  
(msl.equipment.resources.picotech.picoscope.ps6000.Pico  
attribute), 93

MAX\_ANALOGUE\_OFFSET\_5V\_20V  
(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
attribute), 83

MAX\_ANALOGUE\_OFFSET\_5V\_20V  
(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
attribute), 83

MAX\_ANALOGUE\_OFFSET\_5V\_20V  
(msl.equipment.resources.picotech.picoscope.ps4000a.Pico  
attribute), 89

MAX\_ANALOGUE\_OFFSET\_5V\_20V  
(msl.equipment.resources.picotech.picoscope.ps5000a.Pico  
attribute), 92

MAX\_ANALOGUE\_OFFSET\_5V\_20V  
(msl.equipment.resources.picotech.picoscope.ps6000.Pico  
attribute), 93

MAX\_CHANNELS  
(msl.equipment.resources.picotech.picoscope.enums.PS4000Channels  
attribute), 18

MAX\_CHANNELS  
(msl.equipment.resources.picotech.picoscope.enums.PS4000Range  
attribute), 14

(msl.equipment.resources.picotech.picoscope.enums.PS1000AChannel attribute), 28	(msl.equipment.resources.picotech.picoscope.ps2000a.Pico attribute), 82
MAX_CHANNELS	MAX_LOGIC_LEVEL
(msl.equipment.resources.picotech.picoscope.enums.PS1000GChannel attribute), 24	(msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 85
MAX_CHANNELS	MAX_OVERSAMPLE
(msl.equipment.resources.picotech.picoscope.enums.PS1000AChannel attribute), 41	(msl.equipment.resources.picotech.picoscope.ps2000.Pico attribute), 81
MAX_CHANNELS	MAX_OVERSAMPLE
(msl.equipment.resources.picotech.picoscope.enums.PS1000GChannel attribute), 35	(msl.equipment.resources.picotech.picoscope.ps3000.Pico attribute), 84
MAX_CHANNELS	MAX_OVERSAMPLE
(msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 55	(msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 85
MAX_CHANNELS	MAX_OVERSAMPLE_12BIT
(msl.equipment.resources.picotech.picoscope.enums.PS5000GChannel attribute), 50	(msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87
MAX_CHANNELS	MAX_OVERSAMPLE_8BIT
(msl.equipment.resources.picotech.picoscope.enums.PS4000GChannel attribute), 60	(msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87
MAX_DELAY_COUNT	MAX_OVERSAMPLE_8BIT
(msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87	(msl.equipment.resources.picotech.picoscope.ps5000.Pico attribute), 90
MAX_DELAY_COUNT	MAX_OVERSAMPLE_8BIT
(msl.equipment.resources.picotech.picoscope.ps4000a.Pico attribute), 89	(msl.equipment.resources.picotech.picoscope.ps6000.Pico attribute), 93
MAX_DELAY_COUNT	MAX_PROBES (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel attribute), 91
MAX_DELAY_COUNT	MAX_PULSE_WIDTH
(msl.equipment.resources.picotech.picoscope.ps5000a.Pico attribute), 91	(msl.equipment.resources.thorlabs.kinesis.filter_flipper.Filter attribute), 125000A
MAX_ETS_CYCLES	MAX_PULSE_WIDTH_QUALIFIER_COUNT
(msl.equipment.resources.picotech.picoscope.ps6000a.Pico attribute), 93	(msl.equipment.resources.picotech.picoscope.ps3000.Pico attribute), 86000
MAX_ETS_CYCLES_INTERLEAVE_RATIO	MAX_PULSE_WIDTH_QUALIFIER_COUNT
(msl.equipment.resources.picotech.picoscope.ps2000a.Pico attribute), 81	(msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 82000
MAX_ETS_CYCLES_INTERLEAVE_RATIO	MAX_PULSE_WIDTH_QUALIFIER_COUNT
(msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 84	(msl.equipment.resources.picotech.picoscope.ps4000a.Pico attribute), 89000
MAX_EXTRA_RESISTANCES	MAX_PULSE_WIDTH_QUALIFIER_COUNT
(msl.equipment.resources.picotech.picoscope.enums.PS4000Range attribute), 36	(msl.equipment.resources.picotech.picoscope.ps5000.Pico attribute), 86000
MAX_HOLDOFF_COUNT	MAX_PULSE_WIDTH_QUALIFIER_COUNT
(msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 84	(msl.equipment.resources.picotech.picoscope.ps5000a.Pico attribute), 93000
MAX_INTERLEAVE	MAX_PULSE_WIDTH_QUALIFIER_COUNT
(msl.equipment.resources.picotech.picoscope.ps6000a.Pico attribute), 93	(msl.equipment.resources.picotech.picoscope.ps6000.Pico attribute), 96000
MAX_LOGIC_LEVEL	MAX_RANGES (msl.equipment.resources.picotech.picoscope.enums.PS4000Range attribute), 35

MAX_RESISTANCES (msl.equipment.resources.picotech.picoscope.enums.PS4000Range attribute), 35	MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps5000.Pico attribute), 91
MAX_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps2000a.Pico attribute), 82	MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps5000a.Pico attribute), 92
MAX_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 85	MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps6000.Pico attribute), 93
MAX_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87	MAX_TEMPERATURES (msl.equipment.resources.picotech.picoscope.enums.PS4000 attribute), 36
MAX_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps4000a.Pico attribute), 89	MAX_TIMEBASE (msl.equipment.resources.picotech.picoscope.ps2000.Pico attribute), 81
MAX_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps5000.Pico attribute), 91	MAX_TRANSIT_TIME (msl.equipment.resources.thorlabs.kinesis.filter_flipper.Filter attribute), 123
MAX_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps6000.Pico attribute), 93	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS2000 attribute), 18
MAX_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps2000a.Pico attribute), 82	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS3000 attribute), 28
MAX_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 85	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS3000 attribute), 25
MAX_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS4000 attribute), 41
MAX_SIG_GEN_FREQ_4262 (msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS4000 attribute), 35
MAX_SIGGEN_FREQ (msl.equipment.resources.picotech.picoscope.ps2000.Pico attribute), 81	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 55
MAX_SIGGEN_FREQ (msl.equipment.resources.picotech.picoscope.ps3000.Pico attribute), 84	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS5000 attribute), 50
MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps2000a.Pico attribute), 82	MAX_TRIGGER_SOURCES (msl.equipment.resources.picotech.picoscope.enums.PS6000 attribute), 60
MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps3000a.Pico attribute), 85	MAX_UNITS (msl.equipment.resources.picotech.picoscope.ps2000 attribute), 81
MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps4000.Pico attribute), 87	MAX_UNITS (msl.equipment.resources.picotech.picoscope.ps3000 attribute), 83
MAX_SWEEPS_SHOTS (msl.equipment.resources.picotech.picoscope.ps4000a.Pico attribute), 89	MAX_UNITS_OPENED (msl.equipment.resources.picotech.picoscope.enums.PS2000 attribute), 15
	MAX_UNITS_OPENED (msl.equipment.resources.picotech.picoscope.enums.PS3000 attribute), 15

attribute), 26  
 MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000 attribute), 81  
 MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000 attribute), 84  
 MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000 attribute), 87  
 MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000 attribute), 89  
 MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000 attribute), 90  
 MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope6000 attribute), 93  
 MAX\_VALUE\_16BIT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000 attribute), 91  
 MAX\_VALUE\_8BIT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000 attribute), 91  
 MAX\_WAVEFORMS\_PER\_SECOND (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope6000 attribute), 93  
 maxAcceleration (msl.equipment.resources.thorlabs.kinesis.structs.HubDeviceInfo attribute), 159  
 maxChannels (msl.equipment.resources.thorlabs.kinesis.structs.HubDeviceInfo attribute), 157  
 maxDeceleration (msl.equipment.resources.thorlabs.kinesis.structs.MotionStageAxisParameters attribute), 159  
 maxDiameter (msl.equipment.resources.thorlabs.kinesis.structs.IntelQuipParameters attribute), 163  
 maximum\_value() (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope method), 67  
 maxPosition (msl.equipment.resources.thorlabs.kinesis.structs.MotionStageAxisParameters attribute), 159  
 maxTrackingError (msl.equipment.resources.thorlabs.kinesis.structs.MotionStageAxisParameters attribute), 161  
 maxVelocity (msl.equipment.resources.thorlabs.kinesis.structs.MotionStageAxisParameters attribute), 159  
 maxVelocity (msl.equipment.resources.thorlabs.kinesis.structs.MotionStageAxisParameters attribute), 158  
 maxXdemand (msl.equipment.resources.thorlabs.kinesis.structs.QuipPositionDemandParameters attribute), 172  
 maxYdemand (msl.equipment.resources.thorlabs.kinesis.structs.QuipPositionDemandParameters attribute), 172  
 measurement() (msl.equipment.resources.bentham.benhw32.BenthamEquipment attribute), 9  
 MEM\_FAIL (msl.equipment.resources.picotech.picoscope.ps1000a.PicoScope1000 attribute), 15  
 MEM\_FAIL (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000 attribute), 26  
 MEMORY (msl.equipment.resources.picotech.picoscope.enums.PS4000.PicoScope4000 attribute), 45  
 MEMORY\_MAX\_SAMPLES (msl.equipment.resources.picotech.picoscope.enums.PS4000.PicoScope4000 attribute), 47  
 MEMORY\_MAX\_SAMPLES (msl.equipment.resources.picotech.picoscope.enums.PS4000.PicoScope4000 attribute), 47  
 memory\_segments() (msl.equipment.resources.picotech.picoscope.enums.PS4000.PicoScope4000 attribute), 47  
 message\_queue\_size() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 127  
 message\_queue\_size() (msl.equipment.resources.thorlabs.kinesis.integrated\_step\_integrator.IntegratedStepIntegrator method), 136  
 message\_queue\_size() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid\_driver.KCubeSolenoidDriver method), 149  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000 attribute), 82  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000 attribute), 85  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000 attribute), 89  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000 attribute), 92  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope6000 attribute), 92  
 MIN\_ANALOGUE\_OFFSET\_500MV\_200MV (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000 attribute), 82  
 MIN\_ANALOGUE\_OFFSET\_500MV\_200MV (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000 attribute), 85  
 MIN\_ANALOGUE\_OFFSET\_500MV\_200MV (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000 attribute), 89  
 MIN\_ANALOGUE\_OFFSET\_500MV\_200MV (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000 attribute), 92  
 MIN\_ANALOGUE\_OFFSET\_500MV\_200MV (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope6000 attribute), 92





MOT\_CurrentLoopPhases (class in (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
msl.equipment.resources.thorlabs.kinesis.enums), attribute), 109  
110 MOT\_LimitSwitchBreakOnHomeSwapped

MOT\_CustomMotor (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MotorTypes  
attribute), 107 MOT\_LimitSwitchDirectionUndefined

MOT\_DC\_PIDParameters (class in (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_H  
msl.equipment.resources.thorlabs.kinesis.structs), attribute), 108  
162 MOT\_LimitSwitchIgnored

MOT\_DCMotor (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MotorTypes  
attribute), 107 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
attribute), 109

MOT\_DirectionSense (class in MOT\_LimitSwitchIgnored\_Rotational  
msl.equipment.resources.thorlabs.kinesis.enums), (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
108 attribute), 109

MOT\_ForwardLimitSwitch MOT\_LimitSwitchIgnoreSwitch  
(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_EquipmentSwitchDirection  
attribute), 108 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
attribute), 109

MOT\_Forwards (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_DrumSMOCHInvertDirectionSwapped  
attribute), 108 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
attribute), 109

MOT\_HomeLimitSwitchDirection (class in attribute), 109  
msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchMakeOnContact

MOT\_HomingParameters (class in attribute), 109  
msl.equipment.resources.thorlabs.kinesis.structs.MOT\_LimitSwitchMakeOnContactSwapped

MOT\_Immediate (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_I(StopModes  
attribute), 108 attribute), 109 MOT\_LimitSwitchMakeOnHome

MOT\_JogMode (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_ButtonModes.thorlabs.kinesis.enums.MOT\_L  
attribute), 108 attribute), 109

MOT\_JogModes (class in MOT\_LimitSwitchMakeOnHomeSwapped  
msl.equipment.resources.thorlabs.kinesis.enums), (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
108 attribute), 109

MOT\_JogModeUndefined MOT\_LimitSwitchModes (class in  
(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_JogModes.thorlabs.kinesis.enums),  
attribute), 108 109

MOT\_JogParameters (class in MOT\_LimitSwitchModeUndefined  
msl.equipment.resources.thorlabs.kinesis.structs), (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
158 attribute), 109

MOT\_JoystickParameters (class in MOT\_LimitSwitchParameters (class in  
msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.structs),  
160 162

MOT\_LimitsSoftwareApproachPolicy (class in MOT\_LimitSwitchStopImmediate  
msl.equipment.resources.thorlabs.kinesis.enums), (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_L  
109 attribute), 109

MOT\_LimitSwitchBreakOnContact MOT\_LimitSwitchStopImmediate\_Rotational  
(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes.thorlabs.kinesis.enums.MOT\_L  
attribute), 109 attribute), 109

MOT\_LimitSwitchBreakOnContactSwapped MOT\_LimitSwitchStopProfiled  
(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes.thorlabs.kinesis.enums.MOT\_L  
attribute), 109 attribute), 109

MOT\_LimitSwitchBreakOnHome MOT\_LimitSwitchStopProfiled\_Rotational

(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_Profile attribute), 109

MOT\_Profile (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_Profile attribute), 108

MOT\_LimitSwitchSWModes (class in MOT\_Profile attribute), 108

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchSWModes attribute), 108

MOT\_LimitSwitchSWModeUndefined MOT\_ReverseLimitSwitch

(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchSWModes attribute), 108

MOT\_Linear (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_Linear attribute), 107

MOT\_LinearTravelModes (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LinearTravelModes attribute), 108

MOT\_MotorTypes (class in MOT\_SCurve (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MotorTypes attribute), 109

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MotorTypes attribute), 107

MOT\_Normal (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_Normal attribute), 108

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_Normal attribute), 108

MOT\_NotMotor (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_NotMotor attribute), 107

msl.equipment.resources.thorlabs.kinesis.structs.MOT\_NotMotor attribute), 159

MOT\_PhaseA (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PhaseA attribute), 110

MOT\_PhaseA\_CurrentLoopPhases (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PhaseA\_CurrentLoopPhases attribute), 107

MOT\_PhaseAB (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PhaseAB attribute), 110

MOT\_PhaseAB\_CurrentLoopPhases (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PhaseAB\_CurrentLoopPhases attribute), 108

MOT\_PhaseB (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PhaseB attribute), 110

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PhaseB attribute), 108

MOT\_PID\_LoopMode (class in MOT\_StopModeUndefined (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PID\_LoopMode attribute), 108

110 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PID\_LoopMode attribute), 108

MOT\_PIDClosedLoopMode MOT\_Trapezoidal

(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PIDClosedLoopMode attribute), 110

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PIDClosedLoopMode attribute), 109

MOT\_PIDLoopEncoderParams (class in MOT\_TravelDirection (class in msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.enums), 166 108

MOT\_PIDLoopModeDisabled MOT\_TravelDirectionDisabled

(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PIDLoopModeDisabled attribute), 110

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PIDLoopModeDisabled attribute), 108

MOT\_PIDOpenLoopMode MOT\_TravelModes (class in msl.equipment.resources.thorlabs.kinesis.enums), 110 107

MOT\_PMD\_Reserved MOT\_TravelModeUndefined

(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PMD\_Reserved attribute), 109

msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PMD\_Reserved attribute), 107

MOT\_PotentiometerStep (class in MOT\_VelocityParameters (class in msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.structs), 168 158

MOT\_PotentiometerSteps (class in MOT\_VelocityProfileModes (class in msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.enums), 168 109

MOT\_PowerParameters (class in MOT\_VelocityProfileParameters (class in msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.structs), 162 159

MOT\_Preset (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_Preset attribute), 108

msl.equipment.resources.thorlabs.kinesis.structs.MOT\_Preset attribute), 159

MotionControl.MOT\_ButtonMode (class in MotionControl.MOT\_ButtonMode attribute), 108



- msl.equipment.resources.thorlabs.kinesis.motor\_type (module), 7
- 154
- msl.equipment.config (module), 180
- MotionControlCallback (in module msl.equipment.connection (module), 182
- msl.equipment.resources.thorlabs.kinesis.callbacks), 184
- 106
- msl.equipment.connection\_demo (module), 184
- msl.equipment.connection\_msl (module), 184
- motorSignalBias (msl.equipment.resources.thorlabs.kinesis.motor\_type.ElectronicOutputParameters attribute), 162
- motorSignalLimit (msl.equipment.resources.thorlabs.kinesis.motor\_type.ElectronicOutputParameters attribute), 162
- motorType (msl.equipment.resources.thorlabs.kinesis.motor\_type.DeviceTypes (module), 192
- attribute), 157
- msl.equipment.constants (module), 188
- msl.equipment.factory (module), 191
- msl.equipment.resources (module), 8
- move\_absolute() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers\_bentham.IntegratedStepperMotors method), 136
- msl.equipment.resources.bentham.benhw32 (module), 9
- move\_at\_velocity() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers\_bentham.IntegratedStepperMotors method), 136
- msl.equipment.resources.bentham.benhw32 (module), 11
- move\_jog() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers\_bentham.IntegratedStepperMotors method), 136
- msl.equipment.resources.bentham.benhw32 (module), 12
- move\_relative() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers\_bentham.IntegratedStepperMotors method), 136
- msl.equipment.resources.bentham.benhw32 (module), 12
- move\_relative\_distance() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers\_bentham.IntegratedStepperMotors method), 136
- msl.equipment.resources.picotech (module), 12
- msl.equipment.resources.picotech.picoscope (module), 12
- move\_to\_position() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 124
- msl.equipment.resources.picotech.picoscope.callbacks (module), 12
- move\_to\_position() (msl.equipment.resources.thorlabs.kinesis.integrated\_steppers\_bentham.IntegratedStepperMotors method), 136
- msl.equipment.resources.picotech.picoscope.errors (module), 14
- movePercentage (msl.equipment.resources.thorlabs.kinesis.motor\_type.ElectronicOutputParameters attribute), 162
- msl.equipment.resources.picotech.picoscope.errors (module), 64
- MS (msl.equipment.resources.picotech.picoscope.enums.PS2000ATTimeUnits (module), 64
- attribute), 21
- msl.equipment.resources.picotech.picoscope.helper (module), 64
- MS (msl.equipment.resources.picotech.picoscope.enums.PS2000TimeUnits (module), 64
- attribute), 15
- msl.equipment.resources.picotech.picoscope.picoscope (module), 65
- MS (msl.equipment.resources.picotech.picoscope.enums.PS3000ATTimeUnits (module), 65
- attribute), 31
- msl.equipment.resources.picotech.picoscope.picoscope\_2k3k (module), 69
- MS (msl.equipment.resources.picotech.picoscope.enums.PS3000TimeUnits (module), 69
- attribute), 25
- msl.equipment.resources.picotech.picoscope.picoscope\_api (module), 71
- MS (msl.equipment.resources.picotech.picoscope.enums.PS4000ATTimeUnits (module), 71
- attribute), 43
- msl.equipment.resources.picotech.picoscope.ps2000 (module), 80
- MS (msl.equipment.resources.picotech.picoscope.enums.PS4000TimeUnits (module), 80
- attribute), 37
- msl.equipment.resources.picotech.picoscope.ps2000a (module), 82
- MS (msl.equipment.resources.picotech.picoscope.enums.PS5000ATTimeUnits (module), 82
- attribute), 56
- msl.equipment.resources.picotech.picoscope.ps3000 (module), 83
- MS (msl.equipment.resources.picotech.picoscope.enums.PS5000TimeUnits (module), 83
- attribute), 51
- msl.equipment.resources.picotech.picoscope.ps3000a (module), 85
- MS (msl.equipment.resources.picotech.picoscope.enums.PS6000ATTimeUnits (module), 85
- attribute), 61
- MSL (msl.equipment.constants.Backend attribute), 188
- msl.equipment.resources.picotech.picoscope.ps4000 (module), 87

msl.equipment.resources.picotech.picoscope.ps4000a (module), 89  
 msl.equipment.resources.picotech.picoscope.ps5000 (module), 90  
 msl.equipment.resources.picotech.picoscope.ps5000multi\_park() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 msl.equipment.resources.picotech.picoscope.ps6000multi\_select\_wavelength() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 msl.equipment.resources.picotech.picoscope.structs (module), 94  
 msl.equipment.resources.thorlabs (module), 106  
 msl.equipment.resources.thorlabs.fw102c (module), 175  
 msl.equipment.resources.thorlabs.kinesis (module), 106  
 msl.equipment.resources.thorlabs.kinesis.api\_functions (module), 106  
 msl.equipment.resources.thorlabs.kinesis.callbacks (module), 106  
 msl.equipment.resources.thorlabs.kinesis.enums (module), 107  
 msl.equipment.resources.thorlabs.kinesis.errors (module), 123  
 msl.equipment.resources.thorlabs.kinesis.filter\_flipper (module), 123  
 msl.equipment.resources.thorlabs.kinesis.integrated\_reporter (module), 127  
 msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid (module), 146  
 msl.equipment.resources.thorlabs.kinesis.messages (module), 153  
 msl.equipment.resources.thorlabs.kinesis.motion\_controller (module), 154  
 msl.equipment.resources.thorlabs.kinesis.structs (module), 157  
 msl.equipment.resources.utils (module), 178  
 MSLInterface (class in msl.equipment.constants), 188  
 multi\_auto\_measure() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 multi\_auto\_range() (msl.equipment.resources.bentham.benhw32.Bentham32), 9  
 multi\_get\_no\_of\_dark\_currents() (msl.equipment.resources.bentham.benhw32.Bentham32), 9  
 multi\_get\_zero\_calibration\_info() (msl.equipment.resources.bentham.benhw32.Bentham32), 9  
 multi\_initialise() (msl.equipment.resources.bentham.benhw32.Bentham32), 9  
 multi\_measurement() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 multi\_park() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 multi\_select\_wavelength() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 multi\_zero\_calibration() (msl.equipment.resources.bentham.benhw32.Bentham32), 10  
 needs\_homing() (msl.equipment.resources.thorlabs.kinesis.integrated\_reporter), 137  
 NEGATIVE\_RUNT (msl.equipment.resources.picotech.picoscope.enums.PS2000), 23  
 NEGATIVE\_RUNT (msl.equipment.resources.picotech.picoscope.enums.PS3000), 33  
 NEGATIVE\_RUNT (msl.equipment.resources.picotech.picoscope.enums.PS4000), 45  
 NEGATIVE\_RUNT (msl.equipment.resources.picotech.picoscope.enums.PS4000), 39  
 NEGATIVE\_RUNT (msl.equipment.resources.picotech.picoscope.enums.PS5000), 58  
 NEGATIVE\_RUNT (msl.equipment.resources.picotech.picoscope.enums.PS6000), 63  
 newRange (msl.equipment.resources.thorlabs.kinesis.structs.NT\_TRIGGER\_PROPERTIES), 164  
 NO\_OF\_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS4000), 47  
 NO\_OF\_CHANNELS (msl.equipment.resources.picotech.picoscope.picoscope\_a), 75  
 NO\_OF\_TRIGGER\_PROPERTIES (msl.equipment.resources.picotech.picoscope.enums.PS4000), 47  
 NO\_PRESS (msl.equipment.resources.picotech.picoscope.enums.PS4000), 16  
 NONE (msl.equipment.constants.MSLInterface), 189  
 NONE (msl.equipment.constants.Parity attribute), 189



(msl.equipment.resources.thorlabs.kinesis.structs.QD\_NotchFilterParameters attribute), 172

NT\_AutoRangeAtSelected (msl.equipment.resources.thorlabs.kinesis.enums.NT\_AutoRangeAtSelected attribute), 112

NotchFilterOff (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_NotchFilterStates (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 115

NotchFilterOn (msl.equipment.resources.thorlabs.kinesis.enums.BadSig\_PPC\_NotchFilterStates (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 115

notchFilterQ (msl.equipment.resources.thorlabs.kinesis.structs.BNC\_QD\_VLoopParameters (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 171

notchFilterQ (msl.equipment.resources.thorlabs.kinesis.structs.BNC\_QD\_NotchFilterParameters (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 172

notes (msl.equipment.resources.thorlabs.kinesis.structs.NTI\_Info2 (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 158

notUsed (msl.equipment.resources.thorlabs.kinesis.structs.NTI\_EMOT5B (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 161

notUsed (msl.equipment.resources.thorlabs.kinesis.structs.NTI\_EMOT5B (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA attribute), 162

notUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessElectricOutputParameters (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_BI attribute), 161

notUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessPositionLoopParameters (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_BI attribute), 161

notUsed (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessSettleParameters (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_BI attribute), 161

notUsed (msl.equipment.resources.thorlabs.kinesis.structs.NTI\_CMOTAValsyncProfileParameters (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 159

notYetInUse (msl.equipment.resources.thorlabs.kinesis.structs.NTI\_IOSettings (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 165

notYetInUse (msl.equipment.resources.thorlabs.kinesis.structs.NTI\_CircleDiameterLUT (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 174

NS (msl.equipment.resources.picotech.picoscope.enums.NI\_PS2000AAttenuationMode (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 21

NS (msl.equipment.resources.picotech.picoscope.enums.PS2000TimeUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 15

NS (msl.equipment.resources.picotech.picoscope.enums.PS3000AAttenuation (class in msl.equipment.resources.thorlabs.kinesis.structs), attribute), 31

NS (msl.equipment.resources.picotech.picoscope.enums.NI\_PS3000TimeUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 25

NS (msl.equipment.resources.picotech.picoscope.enums.NI\_PS4000AAttenuation (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 43

NS (msl.equipment.resources.picotech.picoscope.enums.PS4000TimeUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 37

NS (msl.equipment.resources.picotech.picoscope.enums.PS5000AAttenuation (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 56

NS (msl.equipment.resources.picotech.picoscope.enums.NI\_PS5000TimeUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 51

NS (msl.equipment.resources.picotech.picoscope.enums.PS6000TimeUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 61

NT\_AbsPowerCircleMode (msl.equipment.resources.thorlabs.kinesis.enums.NT\_CircleDiameterMode (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 112

NT\_Amps (msl.equipment.resources.thorlabs.kinesis.enums.NTI\_PowerInputLimits (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_CubeCircleAdjustment attribute), 113

NT\_AutoRangeAtParameter (msl.equipment.resources.thorlabs.kinesis.enums.NTI\_AutoRangeAtParameter (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_CubeCircleAdjustment attribute), 113

attribute), 113

NT\_CurrentLimit\_500mA (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_CurrentLimit attribute), 113

NT\_Db (msl.equipment.resources.thorlabs.kinesis.enums.NT\_PosiveEquipmentUnits attribute), 113

NT\_Even (msl.equipment.resources.thorlabs.kinesis.enums.NT\_IsolatedEven attribute), 111

NT\_FeedbackSignalAC (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPassFeedbackSignalSelection attribute), 113

NT\_FeedbackSignalDC (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPassFeedbackSignalSelection attribute), 113

NT\_FeedbackSource (class in msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPassFrequency (class in msl.equipment.resources.thorlabs.kinesis.enums), 110

NT\_FeedbackSourceUndefined (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPassFeedbackSource attribute), 111

NT\_FullRange (msl.equipment.resources.thorlabs.kinesis.enums.NT\_SMA2Units attribute), 113

NT\_GainParameters (class in msl.equipment.resources.thorlabs.kinesis.structs), 165

NT\_GoodSignal (msl.equipment.resources.thorlabs.kinesis.enums.NT\_GoodSignal attribute), 110

NT\_HorizontalTracking (msl.equipment.resources.thorlabs.kinesis.enums.NT\_ManualRangeAtSelected attribute), 110

NT\_HubOrSMA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_ModeVoltageRoute attribute), 113

NT\_HVComponent (class in msl.equipment.resources.thorlabs.kinesis.structs), 163

NT\_InRange (msl.equipment.resources.thorlabs.kinesis.enums.NT\_InRange attribute), 111

NT\_IOSettings (class in msl.equipment.resources.thorlabs.kinesis.structs), 164

NT\_Latch (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Mode attribute), 110

NT\_LinearCircleAdjustment (msl.equipment.resources.thorlabs.kinesis.enums.NT\_CircleAdjustment attribute), 112

NT\_LogCircleAdjustment (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LinearCircleAdjustment attribute), 112

NT\_LowPass\_100Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OddLowPassFrequency attribute), 112

NT\_LowPass\_10Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPass attribute), 112

NT\_LowPass\_1Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPass attribute), 112

NT\_LowPass\_300Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPass attribute), 112

NT\_LowPassFeedbackSignalSelection (class in msl.equipment.resources.thorlabs.kinesis.structs), 164

NT\_LowPassFrequency (class in msl.equipment.resources.thorlabs.kinesis.enums), 112

NT\_LowPassFeedbackSource (msl.equipment.resources.thorlabs.kinesis.enums.NT\_LowPassFeedbackSource attribute), 111

NT\_ManualRangeAtParameter (msl.equipment.resources.thorlabs.kinesis.enums.NT\_ManualRangeAtParameter attribute), 112

NT\_ManualRangeAtSelected (msl.equipment.resources.thorlabs.kinesis.enums.NT\_ManualRangeAtSelected attribute), 112

NT\_ModeVoltageRoute (class in msl.equipment.resources.thorlabs.kinesis.enums), 110

NT\_ModeUndefined (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Mode attribute), 110

NT\_NotUnderOrOver (msl.equipment.resources.thorlabs.kinesis.enums.NT\_NotUnderOrOver attribute), 111

NT\_Odd (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Odd attribute), 111

NT\_OddAndEven (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OddAndEven attribute), 111

NT\_OddOrEven (class in msl.equipment.resources.thorlabs.kinesis.enums), 110

NT\_OpenLoop (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OpenLoop attribute), 110

NT\_OneThirdCircleAdjustment (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OneThirdCircleAdjustment attribute), 110

NT\_OneThirdLinearCircleAdjustment (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OneThirdLinearCircleAdjustment attribute), 110

NT\_OneThirdLowPassFrequency (msl.equipment.resources.thorlabs.kinesis.enums.BNT\_OneThirdLowPassFrequency attribute), 112

attribute), 113

NT\_OutputFilter\_10Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAOutputFilter attribute), 113

NT\_OutputFilter\_5kHz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAOutputFilter attribute), 113

NT\_OutputFilter\_None (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAOutputFilter attribute), 113

NT\_OutputVoltageRoute (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 111

NT\_OverRange (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAUnderOrOver attribute), 112

NT\_ParameterCircleMode (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAParameterMode attribute), 112

NT\_Piezo (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAPiezo attribute), 110

NT\_PowerInputUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 113

NT\_SignalState (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 110

NT\_SMA\_Units (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 113

NT\_SMAOnly (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAOutputVoltageRoute attribute), 113

NT\_SquareCircleAdjustment (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIACircleAdjustment attribute), 112

NT\_TIA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIAFeedbackSource attribute), 111

NT\_TIARange (class in msl.equipment.resources.thorlabs.kinesis.enums), NT\_TIARangeParameters (class in msl.equipment.resources.thorlabs.kinesis.structs), 111

NT\_TIARange10\_100uA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange11\_300uA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange12\_1mA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange13\_3mA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange14\_10mA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113

NT\_TIARange1\_3nA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113

NT\_TIARange2\_10nA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113

NT\_TIARange3\_30nA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113

NT\_TIARange4\_100nA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange5\_300nA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange6\_1uA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_TIARange7\_3uA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113

NT\_TIARange8\_10uA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 110

NT\_TIARange9\_30uA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113

NT\_TIAReading (class in msl.equipment.resources.thorlabs.kinesis.structs), 164

NT\_Tracking (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 110

NT\_UnderOrOver (class in msl.equipment.resources.thorlabs.kinesis.enums), 111

NT\_UnderRange (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 111

NT\_UNDEFINERANGE (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange attribute), 113



method), 76

open\_unit\_async\_ex()  
(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
method), 88

open\_unit\_ex()  
(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
method), 88

open\_unit\_progress()  
(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
method), 70

open\_unit\_progress()  
(msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi  
method), 76

openLoopOption (msl.equipment.resources.thorlabs.kinesis.structs.PositionDemandParameters  
attribute), 172

openTime (msl.equipment.resources.thorlabs.kinesis.structs.SetCycleParameters  
attribute), 173

Optical (msl.equipment.resources.thorlabs.kinesis.enums.PPCallbackFeedbackSourceDefinition  
attribute), 115

OPTICAL\_SWITCH  
(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
attribute), 36

OR (msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
attribute), 19

OS\_NOT\_SUPPORTED  
(msl.equipment.resources.picotech.picoscope.enums.PS2000PicoScope2000  
attribute), 15

OS\_NOT\_SUPPORTED  
(msl.equipment.resources.picotech.picoscope.enums.PS3000PicoScope3000  
attribute), 26

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS2000PicoScope2000  
attribute), 24

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS2000PicoScope2000  
attribute), 18

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS3000PicoScope3000  
attribute), 34

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS3000PicoScope3000  
attribute), 28

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
attribute), 47

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
attribute), 40

OUT\_OF\_RANGE  
(msl.equipment.resources.picotech.picoscope.enums.PS5000PicoScope5000  
attribute), 59

OUT\_OF\_RANGE

(msl.equipment.resources.picotech.picoscope.enums.PS5000PicoScope5000  
attribute), 54

(msl.equipment.resources.thorlabs.fw102c.TriggerMode  
attribute), 175

(msl.equipment.resources.thorlabs.kinesis.enums.PS2000PicoScope2000  
attribute), 23

(msl.equipment.resources.picotech.picoscope.enums.PS4000PicoScope4000  
attribute), 39

(msl.equipment.resources.thorlabs.kinesis.enums.PS2000PicoScope2000  
attribute), 53

(msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT  
attribute), 165

(msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT  
attribute), 165

(msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT  
attribute), 164

(msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT  
attribute), 164

(msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT  
attribute), 164

(msl.equipment.resources.thorlabs.kinesis.structs.M  
attribute), 162

(class in msl.equipment.constants), 189

(msl.equipment.resources.thorlabs.kinesis.structs.M  
attribute), 187

(msl.equipment.resources.bentham.benhw32.Bentham32



method), 10  
 park() (msl.equipment.resources.bentham.benhw64.Bentham method), 11  
 parse\_pico\_scope\_api\_header() (in module msl.equipment.resources.picotech.picoscope.ps5000), msl.equipment.resources.picotech.picoscope.helper), 90  
 partNumber (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_EquipAxisParameters attribute), 159  
 path (msl.equipment.config.Config attribute), 182  
 path (msl.equipment.database.Database attribute), 190  
 PENDING (msl.equipment.resources.picotech.picoscope.enums.PS2000OpenProgress attribute), 16  
 PENDING (msl.equipment.resources.picotech.picoscope.enums.PS3000OpenProgress attribute), 26  
 persist\_settings() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.Flipper method), 124  
 persist\_settings() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 137  
 persist\_settings() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150  
 phase (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessCurrentLoopParameters attribute), 161  
 PICOPP\_TOO\_OLD (msl.equipment.resources.picotech.picoscope.enums.PS1000Error attribute), 15  
 PICOPP\_TOO\_OLD (msl.equipment.resources.picotech.picoscope.enums.PS3000Error attribute), 26  
 PicoScope (class in msl.equipment.resources.picotech.picoscope), 66  
 PicoScope2000 (class in msl.equipment.resources.picotech.picoscope.ps2000), 80  
 PicoScope2000A (class in msl.equipment.resources.picotech.picoscope.ps2000a), 82  
 PicoScope2k3k (class in msl.equipment.resources.picotech.picoscope.picoscope\_2k3k), 69  
 PicoScope3000 (class in msl.equipment.resources.picotech.picoscope.ps3000), 83  
 PicoScope3000A (class in msl.equipment.resources.picotech.picoscope.ps3000a), 85  
 PicoScope4000 (class in msl.equipment.resources.picotech.picoscope.ps4000), 87  
 PicoScope4000A (class in msl.equipment.resources.picotech.picoscope.ps4000a), 89  
 PicoScope5000 (class in msl.equipment.resources.picotech.picoscope.ps5000), 91  
 PicoScope5000A (class in msl.equipment.resources.picotech.picoscope.ps5000a), 93  
 PicoScopePS2000OpenProgress (class in msl.equipment.resources.picotech.picoscope.picoscope\_aps2000openprogress), 16  
 PicoScopeChannel (class in msl.equipment.resources.picotech.picoscope.channel), 12  
 PicoScopeIntegrator (class in msl.equipment.resources.picotech.picoscope.enums), 137  
 PID (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_DeviceParameters attribute), 165  
 PIDConstsD (msl.equipment.resources.thorlabs.kinesis.structs.PPC\_PIDConsts attribute), 166  
 PIDConstsI (msl.equipment.resources.thorlabs.kinesis.structs.PPC\_PIDConsts attribute), 166  
 PIDConstsP (msl.equipment.resources.thorlabs.kinesis.structs.PPC\_PIDConsts attribute), 166  
 PIDDescribeFilterOn (msl.equipment.resources.thorlabs.kinesis.structs.PIDDescribeFilterOn attribute), 166  
 PIDOutputLimit (msl.equipment.resources.thorlabs.kinesis.structs.PIDOutputLimit attribute), 167  
 PIDTolerance (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_EquipAxisParameters attribute), 167  
 polarity1 (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_Trip attribute), 170  
 polarity2 (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_Trip attribute), 170  
 ps1000\_duration() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 126  
 ps3000\_duration() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 137  
 ps4000\_duration() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

PORT0 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalPort  
 attribute), 19 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_P  
 PORT0 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort  
 attribute), 29 power1 (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_Trig  
 PORT1 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalPort  
 attribute), 19 power2 (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_Trig  
 PORT1 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort  
 attribute), 29 PPC\_DerivFilterState (class in  
 PORT2 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalPort  
 attribute), 19 resources.thorlabs.kinesis.enums),  
 114  
 PORT2 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort (class in  
 attribute), 29 msl.equipment.resources.thorlabs.kinesis.enums),  
 PORT3 (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalPort  
 attribute), 19 PPC\_IOControlMode (class in  
 PORT3 (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort  
 attribute), 29 resources.thorlabs.kinesis.enums),  
 115  
 PosCorrected (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IOControlMode (class in  
 attribute), 115 msl.equipment.resources.thorlabs.kinesis.enums),  
 posDifference (msl.equipment.resources.thorlabs.kinesis.structs.IQ\_Readings  
 attribute), 172 PPC\_IOOutputBandwidth (class in  
 Position (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_BusEquipmentMode resources.thorlabs.kinesis.enums),  
 attribute), 121 115  
 Position1 (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IOOutputMode (class in  
 attribute), 116 msl.equipment.resources.thorlabs.kinesis.enums),  
 Position2 (msl.equipment.resources.thorlabs.kinesis.enums.FFL\_Positions  
 attribute), 116 PPC\_IOSettings (class in  
 positionErrorLimit msl.equipment.resources.thorlabs.kinesis.structs),  
 (msl.equipment.resources.thorlabs.kinesis.structs.MOT6BrushlessPositionLoopParameters  
 attribute), 161 PPC\_NotchFilterChannel (class in  
 POSITIVE\_RUNT msl.equipment.resources.thorlabs.kinesis.enums),  
 (msl.equipment.resources.picotech.picoscope.enums.PS2000AThresholdDirection  
 attribute), 23 PPC\_NotchFilterState (class in  
 POSITIVE\_RUNT msl.equipment.resources.thorlabs.kinesis.enums),  
 (msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection  
 attribute), 33 PPC\_NotchParams (class in  
 POSITIVE\_RUNT msl.equipment.resources.thorlabs.kinesis.structs),  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection  
 attribute), 45 PPC\_PIDConsts (class in  
 POSITIVE\_RUNT msl.equipment.resources.thorlabs.kinesis.structs),  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection  
 attribute), 39 PRBS (msl.equipment.resources.picotech.picoscope.enums.PS2000  
 POSITIVE\_RUNT attribute), 22  
 (msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection  
 attribute), 58 PRBS (msl.equipment.resources.picotech.picoscope.enums.PS3000  
 attribute), 32  
 POSITIVE\_RUNT PRBS (msl.equipment.resources.picotech.picoscope.enums.PS4000  
 (msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdDirection  
 attribute), 63 PRBS (msl.equipment.resources.picotech.picoscope.enums.PS4000  
 PosRaw (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IOOutputMode  
 attribute), 115 PRBS (msl.equipment.resources.picotech.picoscope.enums.PS5000  
 postCycleDelay (msl.equipment.resources.thorlabs.kinesis.structs.IZEL54WaveParameters  
 attribute), 165 PRBS (msl.equipment.resources.picotech.picoscope.enums.PS6000

attribute), 62

PRBS\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_PulseGen), attribute), 83

PRBS\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000a.PulseGen), attribute), 86

PRBS\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps6000a.PulseGen), attribute), 93

PRBS\_MIN\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps2000a.PulseGen), attribute), 83

PRBS\_MIN\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000a.PulseGen), attribute), 86

pre\_trigger (msl.equipment.resources.picotech.picoscope.enums.PicScope), attribute), 66

preCycleDelay (msl.equipment.resources.thorlabs.kinesis.structs.PZnWaveParam), attribute), 165

PresetPos1 (msl.equipment.resources.thorlabs.kinesis.structs.KVIOFF\_MMIParam), attribute), 168

PresetPos1 (msl.equipment.resources.thorlabs.kinesis.structs.KPZnMMIParam), attribute), 170

PresetPos2 (msl.equipment.resources.thorlabs.kinesis.structs.KVIOFF\_MMIParam), attribute), 168

PresetPos2 (msl.equipment.resources.thorlabs.kinesis.structs.KPZnMMIParam), attribute), 170

PRESSURE\_SENSOR\_50BAR (msl.equipment.resources.picotech.picoscope.enums.PS4000Probe), attribute), 36

PRESSURE\_SENSOR\_5BAR (msl.equipment.resources.picotech.picoscope.enums.PS5000Probe), attribute), 36

print\_class\_def\_signatures() (in module msl.equipment.resources.picotech.picoscope.helper), 65

print\_common\_functions() (in module msl.equipment.resources.picotech.picoscope.helper), 65

print\_define\_statements() (in module msl.equipment.resources.picotech.picoscope.helper), 65

PROBES (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelInfo), attribute), 37

PROLOGIX\_ENET (msl.equipment.constants.MSLInterface), attribute), 189

PROLOGIX\_USB (msl.equipment.constants.MSLInterface), attribute), 189

properties (msl.equipment.record\_types.ConnectionRecord), attribute), 194

prop2000aPGain (msl.equipment.resources.thorlabs.kinesis.structs.PS2000A\_PulseGen), attribute), 161

proportionalGain (msl.equipment.resources.thorlabs.kinesis.structs.PS3000a.PulseGen), attribute), 163

proportionalGain (msl.equipment.resources.thorlabs.kinesis.structs.PS6000a.PulseGen), attribute), 167

proportionalGain (msl.equipment.resources.thorlabs.kinesis.structs.PS2000a.PulseGen), attribute), 172

proportionalGain (msl.equipment.resources.thorlabs.kinesis.structs.PS3000a.PulseGen), attribute), 171

proportionalGain (msl.equipment.resources.thorlabs.kinesis.structs.PS6000a.PulseGen), attribute), 175

proportionalGain (msl.equipment.resources.thorlabs.kinesis.structs.PS2000a.PulseGen), attribute), 175

PS (msl.equipment.resources.picotech.picoscope.enums.PS2000Tire), attribute), 21

PS (msl.equipment.resources.picotech.picoscope.enums.PS3000Tire), attribute), 15

PS (msl.equipment.resources.picotech.picoscope.enums.PS3000ATire), attribute), 31

PS (msl.equipment.resources.picotech.picoscope.enums.PS4000ATire), attribute), 43

PS (msl.equipment.resources.picotech.picoscope.enums.PS4000Probe), attribute), 35

PS (msl.equipment.resources.picotech.picoscope.enums.PS5000ATire), attribute), 56

PS (msl.equipment.resources.picotech.picoscope.enums.PS5000Tire), attribute), 51

PS (msl.equipment.resources.picotech.picoscope.enums.PS6000Tire), attribute), 61

PS2000A\_ThresholdMode (class in msl.equipment.resources.picotech.picoscope.enums), 22

PS2000AChannel (class in msl.equipment.resources.picotech.picoscope.enums), 18

PS2000AChannelBufferIndex (class in msl.equipment.resources.picotech.picoscope.enums), 18

PS2000AChannelInfo (class in msl.equipment.resources.picotech.picoscope.enums), 21

PS2000ACoupling (class in msl.equipment.resources.picotech.picoscope.enums), 20

PS2000ADigitalChannel	(class in PS2000ATriggerConditions	(class in msl.equipment.resources.picotech.picoscope.enums),
19	95	
PS2000ADigitalChannelDirections	(class in PS2000ATriggerOperand	(class in msl.equipment.resources.picotech.picoscope.structs),
96	18	
PS2000ADigitalDirection	(class in PS2000ATriggerState	(class in msl.equipment.resources.picotech.picoscope.enums),
23	23	
PS2000AEtsMode	(class in PS2000AWaveType	(class in msl.equipment.resources.picotech.picoscope.enums),
21	21	
PS2000AExtraOperations	(class in PS2000ButtonState	(class in msl.equipment.resources.picotech.picoscope.enums),
22	16	
PS2000AHoldOffType	(class in PS2000Channel	(class in msl.equipment.resources.picotech.picoscope.enums),
24	14	
PS2000AIndexMode	(class in PS2000DigitalPort	(class in msl.equipment.resources.picotech.picoscope.enums),
22	19	
PS2000APulseWidthType	(class in PS2000Error	(class in msl.equipment.resources.picotech.picoscope.enums),
24	15	
PS2000APwqConditions	(class in PS2000EtsMode	(class in msl.equipment.resources.picotech.picoscope.structs),
96	16	
PS2000ARange	(class in PS2000Info	(class in msl.equipment.resources.picotech.picoscope.enums),
20	15	
PS2000ARatioMode	(class in PS2000OpenProgress	(class in msl.equipment.resources.picotech.picoscope.enums),
24	16	
PS2000ASigGenTrigSource	(class in PS2000PulseWidthType	(class in msl.equipment.resources.picotech.picoscope.enums),
22	18	
PS2000ASigGenTrigType	(class in PS2000PwqConditions	(class in msl.equipment.resources.picotech.picoscope.structs),
22	95	
PS2000ASweepType	(class in PS2000Range	(class in msl.equipment.resources.picotech.picoscope.enums),
21	14	
PS2000AThresholdDirection	(class in PS2000SweepType	(class in msl.equipment.resources.picotech.picoscope.enums),
23	16	
PS2000ATimeUnits	(class in PS2000ThresholdDirection	(class in msl.equipment.resources.picotech.picoscope.enums),
21	17	
PS2000ATriggerChannelProperties	(class in PS2000ThresholdMode	(class in msl.equipment.resources.picotech.picoscope.structs),
96	17	

PS2000TimeUnits	(class	in PS2205_MAX_ETS_CYCLES	
15	mssl.equipment.resources.picotech.picoscope.enums),	(mssl.equipment.resources.picotech.picoscope.ps2000.Pico	attribute), 81
PS2000TriggerChannelProperties	(class	in PS2205_MAX_ETS_INTERLEAVE	
94	mssl.equipment.resources.picotech.picoscope.structs),	(mssl.equipment.resources.picotech.picoscope.ps2000.Pico	attribute), 81
PS2000TriggerConditions	(class	in PS2206_MAX_ETS_CYCLES	
95	mssl.equipment.resources.picotech.picoscope.structs),	(mssl.equipment.resources.picotech.picoscope.ps2000a.Pico	attribute), 82
PS2000TriggerDirection	(class	in PS2206_MAX_INTERLEAVE	
16	mssl.equipment.resources.picotech.picoscope.enums),	(mssl.equipment.resources.picotech.picoscope.ps2000a.Pico	attribute), 82
PS2000TriggerState	(class	in PS2207_MAX_ETS_CYCLES	
17	mssl.equipment.resources.picotech.picoscope.enums),	(mssl.equipment.resources.picotech.picoscope.ps2000a.Pico	attribute), 82
PS2000WaveType	(class	in PS2207_MAX_INTERLEAVE	
17	mssl.equipment.resources.picotech.picoscope.enums),	(mssl.equipment.resources.picotech.picoscope.ps2000a.Pico	attribute), 82
PS2104_MAX_ETS_CYCLES		PS2208_MAX_ETS_CYCLES	
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.ps2000a.Pico	attribute), 81
PS2104_MAX_ETS_INTERLEAVE		PS2208_MAX_INTERLEAVE	
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.ps2000a.Pico	attribute), 81
PS2104_MAX_TIMEBASE		PS3000A_ThresholdMode	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2105_MAX_ETS_CYCLES		PS3000ABandwidthLimiter	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2105_MAX_ETS_INTERLEAVE		PS3000AChannel	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2105_MAX_TIMEBASE		PS3000AChannelBufferIndex	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2200_MAX_TIMEBASE		PS3000AChannelInfo	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2203_MAX_ETS_CYCLES		PS3000ACoupling	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2203_MAX_ETS_INTERLEAVE		PS3000ADigitalChannel	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81
PS2204_MAX_ETS_CYCLES		PS3000ADigitalChannelDirections	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.structs),	attribute), 81
PS2204_MAX_ETS_INTERLEAVE		PS3000ADigitalDirection	(class in
	(mssl.equipment.resources.picotech.picoscope.ps2000.PicoEquipme	mssl.equipment.resources.picotech.picoscope.enums),	attribute), 81

PS3000ADigitalPort	(class	in PS3000ATriggerConditionsV2	(class	in
28	msl.equipment.resources.picotech.picoscope.enums),	98	msl.equipment.resources.picotech.picoscope.structs),	
PS3000AEtsMode	(class	in PS3000ATriggerInfo	(class	in
30	msl.equipment.resources.picotech.picoscope.enums),	100	msl.equipment.resources.picotech.picoscope.structs),	
PS3000AExtraOperations	(class	in PS3000ATriggerState	(class	in
32	msl.equipment.resources.picotech.picoscope.enums),	33	msl.equipment.resources.picotech.picoscope.enums),	
PS3000AHoldOffType	(class	in PS3000AWaveType	(class	in
34	msl.equipment.resources.picotech.picoscope.enums),	31	msl.equipment.resources.picotech.picoscope.enums),	
PS3000AIndexMode	(class	in PS3000Channel	(class	in
32	msl.equipment.resources.picotech.picoscope.enums),	24	msl.equipment.resources.picotech.picoscope.enums),	
PS3000APulseWidthType	(class	in PS3000Error	(class	in
34	msl.equipment.resources.picotech.picoscope.enums),	26	msl.equipment.resources.picotech.picoscope.enums),	
PS3000APwqConditions	(class	in PS3000EtsMode	(class	in
98	msl.equipment.resources.picotech.picoscope.structs),	27	msl.equipment.resources.picotech.picoscope.enums),	
PS3000APwqConditionsV2	(class	in PS3000Info	(class	in
99	msl.equipment.resources.picotech.picoscope.structs),	26	msl.equipment.resources.picotech.picoscope.enums),	
PS3000ARange	(class	in PS3000OpenProgress	(class	in
30	msl.equipment.resources.picotech.picoscope.enums),	26	msl.equipment.resources.picotech.picoscope.enums),	
PS3000ARatioMode	(class	in PS3000PulseWidthType	(class	in
34	msl.equipment.resources.picotech.picoscope.enums),	27	msl.equipment.resources.picotech.picoscope.enums),	
PS3000ASigGenTrigSource	(class	in PS3000PwqConditions	(class	in
32	msl.equipment.resources.picotech.picoscope.enums),	97	msl.equipment.resources.picotech.picoscope.structs),	
PS3000ASigGenTrigType	(class	in PS3000Range	(class	in
32	msl.equipment.resources.picotech.picoscope.enums),	25	msl.equipment.resources.picotech.picoscope.enums),	
PS3000ASweepType	(class	in PS3000ThresholdDirection	(class	in
31	msl.equipment.resources.picotech.picoscope.enums),	27	msl.equipment.resources.picotech.picoscope.enums),	
PS3000AThresholdDirection	(class	in PS3000ThresholdMode	(class	in
32	msl.equipment.resources.picotech.picoscope.enums),	27	msl.equipment.resources.picotech.picoscope.enums),	
PS3000ATimeUnits	(class	in PS3000TimeUnits	(class	in
31	msl.equipment.resources.picotech.picoscope.enums),	25	msl.equipment.resources.picotech.picoscope.enums),	
PS3000ATriggerChannelProperties	(class	in PS3000TriggerChannelProperties	(class	in
99	msl.equipment.resources.picotech.picoscope.structs),	97	msl.equipment.resources.picotech.picoscope.structs),	
PS3000ATriggerConditions	(class	in PS3000TriggerConditions	(class	in
98	msl.equipment.resources.picotech.picoscope.structs),	97	msl.equipment.resources.picotech.picoscope.structs),	

PS3000TriggerDirection (class in PS3206\_MAX\_ETS\_INTERLEAVE  
 msl.equipment.resources.picotech.picoscope.enums)(msl.equipment.resources.picotech.picoscope.ps3000.Pico  
 26 attribute), 84

PS3000TriggerState (class in PS3206\_MAX\_TIMEBASE  
 msl.equipment.resources.picotech.picoscope.enums)(msl.equipment.resources.picotech.picoscope.ps3000.Pico  
 27 attribute), 83

PS3000WaveTypes (class in PS3206A\_MAX\_ETS\_CYCLES  
 msl.equipment.resources.picotech.picoscope.enums)(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
 25 attribute), 85

PS300\_MAX\_ETS\_SAMPLES PS3206A\_MAX\_INTERLEAVE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000a.Pico  
 attribute), 84 attribute), 85

PS3204\_MAX\_ETS\_CYCLES PS3206B\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000a.Pico  
 attribute), 84 attribute), 85

PS3204\_MAX\_ETS\_INTERLEAVE PS3206MSO\_MAX\_INTERLEAVE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000a.Pico  
 attribute), 84 attribute), 85

PS3204\_MAX\_TIMEBASE PS3207A\_MAX\_ETS\_CYCLES  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000a.Pico  
 attribute), 83 attribute), 85

PS3204A\_MAX\_ETS\_CYCLES PS3207A\_MAX\_INTERLEAVE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000Aresources.picotech.picoscope.ps3000a.Pico  
 attribute), 85 attribute), 85

PS3204A\_MAX\_INTERLEAVE PS3207B\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000Aresources.picotech.picoscope.ps3000a.Pico  
 attribute), 85 attribute), 85

PS3204MSO\_MAX\_INTERLEAVE PS3223\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000Aresources.picotech.picoscope.ps3000.Pico  
 attribute), 85 attribute), 83

PS3205\_MAX\_ETS\_CYCLES PS3224\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000.Pico  
 attribute), 84 attribute), 83

PS3205\_MAX\_ETS\_INTERLEAVE PS3225\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000.Pico  
 attribute), 84 attribute), 84

PS3205\_MAX\_TIMEBASE PS3226\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000.Pico  
 attribute), 83 attribute), 84

PS3205A\_MAX\_ETS\_CYCLES PS3423\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000Aresources.picotech.picoscope.ps3000.Pico  
 attribute), 85 attribute), 84

PS3205A\_MAX\_INTERLEAVE PS3424\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000Aresources.picotech.picoscope.ps3000.Pico  
 attribute), 85 attribute), 84

PS3205MSO\_MAX\_INTERLEAVE PS3425\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000Aresources.picotech.picoscope.ps3000.Pico  
 attribute), 85 attribute), 84

PS3206\_MAX\_ETS\_CYCLES PS3426\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000resources.picotech.picoscope.ps3000.Pico  
 attribute), 84 attribute), 84

PS4000ABandwidthLimiter	(class in PS4000AMetaType	(class in msl.equipment.resources.picotech.picoscope.enums),
40		44
PS4000AChannel	(class in PS4000APicoStringValue	(class in msl.equipment.resources.picotech.picoscope.enums),
41		47
PS4000AChannelBufferIndex	(class in PS4000APulseWidthType	(class in msl.equipment.resources.picotech.picoscope.enums),
41		46
PS4000AChannelInfo	(class in PS4000ARange	(class in msl.equipment.resources.picotech.picoscope.enums),
47		42
PS4000AChannelLed	(class in PS4000ARatioMode	(class in msl.equipment.resources.picotech.picoscope.enums),
44		46
PS4000AChannelLedSetting	(class in PS4000AResistanceRange	(class in msl.equipment.resources.picotech.picoscope.structs),
101		42
PS4000ACondition	(class in PS4000ASensorState	(class in msl.equipment.resources.picotech.picoscope.structs),
101		46
PS4000AConditionsInfo	(class in PS4000ASigGenTrigSource	(class in msl.equipment.resources.picotech.picoscope.enums),
46		44
PS4000AConnectDetect	(class in PS4000ASigGenTrigType	(class in msl.equipment.resources.picotech.picoscope.structs),
102		44
PS4000ACoupling	(class in PS4000ASweepType	(class in msl.equipment.resources.picotech.picoscope.enums),
41		43
PS4000ADirection	(class in PS4000AThresholdDirection	(class in msl.equipment.resources.picotech.picoscope.structs),
101		45
PS4000AEtsMode	(class in PS4000AThresholdMode	(class in msl.equipment.resources.picotech.picoscope.enums),
42		45
PS4000AExtraOperations	(class in PS4000ATimeUnits	(class in msl.equipment.resources.picotech.picoscope.enums),
40		43
PS4000AFrequencyCounterRange	(class in PS4000ATriggerChannelProperties	(class in msl.equipment.resources.picotech.picoscope.enums),
46		102
PS4000AIndexMode	(class in PS4000ATriggerState	(class in msl.equipment.resources.picotech.picoscope.enums),
45		46
PS4000AMetaFormat	(class in PS4000AWaveType	(class in msl.equipment.resources.picotech.picoscope.enums),
44		43
PS4000AMetaOperation	(class in PS4000Channel	(class in msl.equipment.resources.picotech.picoscope.enums),
44		34



PS4000ChannelBufferIndex	(class in PS4000TimeUnits	(class in msl.equipment.resources.picotech.picoscope.enums),
34		37
PS4000ChannelInfo	(class in PS4000TriggerChannelProperties	(class in msl.equipment.resources.picotech.picoscope.structs),
36		101
PS4000EtsMode	(class in PS4000TriggerConditions	(class in msl.equipment.resources.picotech.picoscope.structs),
37		100
PS4000FrequencyCounterRange	(class in PS4000TriggerState	(class in msl.equipment.resources.picotech.picoscope.enums),
40		39
PS4000IndexMode	(class in PS4000WaveType	(class in msl.equipment.resources.picotech.picoscope.enums),
38		38
PS4000OperationTypes	(class in PS4262_MAX_VALUE	(msl.equipment.resources.picotech.picoscope.ps4000.Pico
37		attribute), 87
PS4000Probe	(class in PS4262_MAX_WAVEFORM_BUFFER_SIZE	(msl.equipment.resources.picotech.picoscope.ps4000.Pico
36		attribute), 87
PS4000Ps4000HoldOffType	(class in PS4262_MIN_DWELL_COUNT	(msl.equipment.resources.picotech.picoscope.ps4000.Pico
40		attribute), 87
PS4000PulseWidthType	(class in PS4262_MIN_VALUE	(msl.equipment.resources.picotech.picoscope.ps4000.Pico
40		attribute), 87
PS4000PwqConditions	(class in PS4XXX_MAX_ETS_CYCLES	(msl.equipment.resources.picotech.picoscope.ps4000.Pico
100		attribute), 87
PS4000Range	(class in PS4XXX_MAX_INTERLEAVE	(msl.equipment.resources.picotech.picoscope.ps4000.Pico
35		attribute), 87
PS4000RatioMode	(class in PS5000ABandwidthLimiter	(class in msl.equipment.resources.picotech.picoscope.enums),
39		54
PS4000SigGenTrigSource	(class in PS5000AChannel	(class in msl.equipment.resources.picotech.picoscope.enums),
38		54
PS4000SigGenTrigType	(class in PS5000AChannelBufferIndex	(class in msl.equipment.resources.picotech.picoscope.enums),
38		55
PS4000SweepType	(class in PS5000AChannelInfo	(class in msl.equipment.resources.picotech.picoscope.enums),
37		59
PS4000ThresholdDirection	(class in PS5000ACoupling	(class in msl.equipment.resources.picotech.picoscope.enums),
39		54
PS4000ThresholdMode	(class in PS5000ADeviceResolution	(class in msl.equipment.resources.picotech.picoscope.enums),
39		54

PS5000AEtsMode	(class	in PS5000ATriggerWithinPreTrigger	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
56		58		
PS5000AExtraOperations	(class	in PS5000AWaveType	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
54		56		
PS5000AIndexMode	(class	in PS5000Channel	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
57		50		
PS5000APulseWidthType	(class	in PS5000ChannelBufferIndex	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
59		50		
PS5000APwqConditions	(class	in PS5000ChannelInfo	(class	in
	mssl.equipment.resources.picotech.picoscope.structs),		mssl.equipment.resources.picotech.picoscope.enums),	
104		54		
PS5000ARange	(class	in PS5000EtsMode	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
55		51		
PS5000ARatioMode	(class	in PS5000IndexMode	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
58		52		
PS5000ASigGenTrigSource	(class	in PS5000PulseWidthType	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
57		53		
PS5000ASigGenTrigType	(class	in PS5000PwqConditions	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.structs),	
57		103		
PS5000ASweepType	(class	in PS5000Range	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
56		50		
PS5000AThresholdDirection	(class	in PS5000RatioMode	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
57		53		
PS5000AThresholdMode	(class	in PS5000SigGenTrigSource	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
57		52		
PS5000ATimeUnits	(class	in PS5000SigGenTrigType	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
56		52		
PS5000ATriggerChannelProperties	(class	in PS5000SweepType	(class	in
	mssl.equipment.resources.picotech.picoscope.structs),		mssl.equipment.resources.picotech.picoscope.enums),	
104		51		
PS5000ATriggerConditions	(class	in PS5000ThresholdDirection	(class	in
	mssl.equipment.resources.picotech.picoscope.structs),		mssl.equipment.resources.picotech.picoscope.enums),	
104		53		
PS5000ATriggerInfo	(class	in PS5000ThresholdMode	(class	in
	mssl.equipment.resources.picotech.picoscope.structs),		mssl.equipment.resources.picotech.picoscope.enums),	
103		52		
PS5000ATriggerState	(class	in PS5000TimeUnits	(class	in
	mssl.equipment.resources.picotech.picoscope.enums),		mssl.equipment.resources.picotech.picoscope.enums),	
58		51		

PS5000TriggerChannelProperties	(class in PS6000EtsMode	(class in
msl.equipment.resources.picotech.picoscope.structs),	msl.equipment.resources.picotech.picoscope.enums),	
103	61	
PS5000TriggerConditions	(class in PS6000ExternalFrequency	(class in
msl.equipment.resources.picotech.picoscope.structs),	msl.equipment.resources.picotech.picoscope.enums),	
102	59	
PS5000TriggerState	(class in PS6000ExtraOperations	(class in
msl.equipment.resources.picotech.picoscope.enums),	msl.equipment.resources.picotech.picoscope.enums),	
53	62	
PS5000WaveType	(class in PS6000IndexMode	(class in
msl.equipment.resources.picotech.picoscope.enums),	msl.equipment.resources.picotech.picoscope.enums),	
51	62	
PS5242A_MAX_ETS_CYCLES	PS6000PulseWidthType	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	64	
PS5242A_MAX_ETS_INTERLEAVE	PS6000PwqConditions	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.structs),	
92	105	
PS5243A_MAX_ETS_CYCLES	PS6000Range	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	60	
PS5243A_MAX_ETS_INTERLEAVE	PS6000RatioMode	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	63	
PS5244A_MAX_ETS_CYCLES	PS6000SigGenTrigSource	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	62	
PS5244A_MAX_ETS_INTERLEAVE	PS6000SigGenTrigType	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	62	
PS5X42A_MAX_SIG_GEN_BUFFER_SIZE	PS6000SweepType	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	61	
PS5X43A_MAX_SIG_GEN_BUFFER_SIZE	PS6000ThresholdDirection	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	63	
PS5X44A_MAX_SIG_GEN_BUFFER_SIZE	PS6000ThresholdMode	(class in
(msl.equipment.resources.picotech.picoscope.ps5000Attribute),	msl.equipment.resources.picotech.picoscope.enums),	
92	63	
PS6000BandwidthLimiter	(class in PS6000TimeUnits	(class in
msl.equipment.resources.picotech.picoscope.enums),	msl.equipment.resources.picotech.picoscope.enums),	
59	61	
PS6000Channel	(class in PS6000TriggerChannelProperties	(class in
msl.equipment.resources.picotech.picoscope.enums),	msl.equipment.resources.picotech.picoscope.structs),	
59	105	
PS6000ChannelBufferIndex	(class in PS6000TriggerConditions	(class in
msl.equipment.resources.picotech.picoscope.enums),	msl.equipment.resources.picotech.picoscope.structs),	
60	105	
PS6000Coupling	(class in PS6000TriggerState	(class in
msl.equipment.resources.picotech.picoscope.enums),	msl.equipment.resources.picotech.picoscope.enums),	
60	63	

PS6000WaveType (class in pulseWidthQualifier  
 msl.equipment.resources.picotech.picoscope.enums), (msl.equipment.resources.picotech.picoscope.structs.PS3000PicoScopePicoString  
 attribute), 98

PS640X\_C\_D\_MAX\_SIG\_GEN\_BUFFER\_SIZE pulseWidthQualifier  
 (msl.equipment.resources.picotech.picoscope.ps6000.PicoScopePicoStringValue), (msl.equipment.resources.picotech.picoscope.structs.PS3000PicoScopePicoString  
 attribute), 93

PULSE\_WIDTH\_ARRAY\_OF\_BLOCK\_CONDITIONS pulseWidthQualifier  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue), (msl.equipment.resources.picotech.picoscope.structs.PS4000PicoScopePicoString  
 attribute), 48

PULSE\_WIDTH\_CONDITIONS pulseWidthQualifier  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue), (msl.equipment.resources.picotech.picoscope.structs.PS5000PicoScopePicoString  
 attribute), 48

PULSE\_WIDTH\_CONDITIONS\_SOURCE pulseWidthQualifier  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue), (msl.equipment.resources.picotech.picoscope.structs.PS5000PicoScopePicoString  
 attribute), 48

PULSE\_WIDTH\_CONDITIONS\_STATE pulseWidthQualifier  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue), (msl.equipment.resources.picotech.picoscope.structs.PS6000PicoScopePicoString  
 attribute), 48

PULSE\_WIDTH\_NO\_OF\_BLOCK\_CONDITIONS PyVISA (msl.equipment.constants.MSLInterface at-  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 48 PyVISA (msl.equipment.constants.Backend  
 attribute), 188

PULSE\_WIDTH\_NO\_OF\_CONDITIONS PyVISA.PS3000PicoStringValue  
 (msl.equipment.resources.picotech.picoscope.enums.PS3000PicoStringValue  
 attribute), 48 (msl.equipment.config.Config attribute),  
 182

PULSE\_WIDTH\_PROPERTIES (msl.equipment.resources.picotech.picoscope.enums.PZ\_Enum), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Input  
 attribute), 48

PULSE\_WIDTH\_PROPERTIES\_DIRECTION PZ\_CloseLoop (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum),  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 48 PZ\_CloseLoopSmooth

PULSE\_WIDTH\_PROPERTIES\_LOWER (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Constant), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Constant),  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 48 PZ\_Continuous (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum),  
 (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), 114

PULSE\_WIDTH\_PROPERTIES\_TYPE (msl.equipment.resources.picotech.picoscope.enums.PZ\_Enum), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum),  
 (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), 114

PULSE\_WIDTH\_PROPERTIES\_UPPER (msl.equipment.resources.picotech.picoscope.enums.PZ\_Enum), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum),  
 (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), 114

PULSE\_WIDTH\_SOURCE (msl.equipment.resources.picotech.picoscope.enums.PZ\_Enum), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum),  
 (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), 114

pulseWidthQualifier (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), (msl.equipment.resources.thorlabs.kinesis.structs),  
 165

pulseWidthQualifier (msl.equipment.resources.picotech.picoscope.enums.PZ\_Fixed), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_OutputTrigEnable),  
 (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), 114

pulseWidthQualifier PZ\_InputSourceFlags (class in  
 (msl.equipment.resources.picotech.picoscope.structs.PS2000PicoStringValue), (msl.equipment.resources.thorlabs.kinesis.enums),  
 attribute), 95 114

pulseWidthQualifier PZ\_InputTrigEnable  
 (msl.equipment.resources.picotech.picoscope.structs.PS3000PicoStringValue), (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_OutputTrigEnable),  
 (msl.equipment.resources.thorlabs.kinesis.enums.PZ\_Enum), 98 attribute), 114



msl.equipment.resources.thorlabs.kinesis.stb.R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS200 attribute), 172  
 QD\_RouteUndefined (msl.equipment.resources.thorlabs.kinesis.enums.QDAttribute), 119  
 QD\_SMAOnly (msl.equipment.resources.thorlabs.kinesis.enums.QDAttribute), 119  
 QD\_Trig\_Disabled (msl.equipment.resources.thorlabs.kinesis.enums.QDAttribute), 120  
 QD\_TrigIn\_GPI (msl.equipment.resources.thorlabs.kinesis.enums.QDAttribute), 120  
 QD\_TrigIn\_LoopOpenClose (msl.equipment.resources.thorlabs.kinesis.enums.QDAttribute), 120  
 QD\_Undefined (msl.equipment.resources.thorlabs.kinesis.enums.QDAttribute), 120  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS200Attribute), 22  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS300Attribute), 32  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS400Attribute), 45  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS400Attribute), 39  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS500Attribute), 57  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS500Attribute), 52  
 QUAD (msl.equipment.resources.picotech.picoscope.enums.PS600Attribute), 62  
 query() (msl.equipment.connection\_msl.ConnectionMessageBus attribute), 15  
 method), 186  
**R**  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS200Attribute), 20  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS200Attribute), 14  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS300Attribute), 30  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS300Attribute), 25  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS400Attribute), 42  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS400Attribute), 35  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS500Attribute), 55  
 R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS500Attribute), 50  
 R\_100V (msl.equipment.resources.picotech.picoscope.enums.PS300 attribute), 60  
 R\_100V (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 125  
 R\_100V (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 142  
 R\_100V (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 35  
 R\_100V (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 42  
 R\_100V (msl.equipment.resources.picotech.picoscope.enums.PS200 attribute), 20  
 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS200 attribute), 120  
 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS300 attribute), 110  
 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS300 attribute), 125  
 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 42  
 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS500 attribute), 55  
 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS600 attribute), 60  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS200 attribute), 15  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS300 attribute), 30  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS300 attribute), 40  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 40  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 50  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS500 attribute), 50  
 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS600 attribute), 60  
 R\_1100K (msl.equipment.resources.picotech.picoscope.enums.PS400 attribute), 35  
 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS200 attribute), 20  
 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS200 attribute), 20



attribute), 25

R\_500MV (msl.equipment.resources.picotech.picoscope.enums.PS3000) Range attribute), 42

R\_500MV (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 35

R\_500MV (msl.equipment.resources.picotech.picoscope.enums.PS5000) Range attribute), 55

R\_500MV (msl.equipment.resources.picotech.picoscope.enums.PS6000) Range attribute), 51

R\_500MV (msl.equipment.resources.picotech.picoscope.enums.PS7000) Range attribute), 60

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS1000) Range attribute), 20

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS2000) Range attribute), 14

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS3000) Range attribute), 30

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 25

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS5000) Range attribute), 42

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS6000) Range attribute), 35

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS7000) Range attribute), 55

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS8000) Range attribute), 50

R\_50MV (msl.equipment.resources.picotech.picoscope.enums.PS9000) Range attribute), 60

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS2000) Range attribute), 20

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS3000) Range attribute), 15

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 30

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS5000) Range attribute), 25

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS6000) Range attribute), 42

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS7000) Range attribute), 35

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS8000) Range attribute), 56

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS9000) Range attribute), 51

R\_50V (msl.equipment.resources.picotech.picoscope.enums.PS10000) Range attribute), 60

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS2000) Range attribute), 20

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS3000) Range attribute), 15

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 20

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS5000) Range attribute), 25

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS6000) Range attribute), 30

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS7000) Range attribute), 35

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS8000) Range attribute), 40

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS9000) Range attribute), 45

R\_5V (msl.equipment.resources.picotech.picoscope.enums.PS10000) Range attribute), 50

R\_ADCV (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 30

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000) Range attribute), 15

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000) Range attribute), 20

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 25

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000) Range attribute), 30

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000) Range attribute), 35

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS7000) Range attribute), 40

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS8000) Range attribute), 45

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS9000) Range attribute), 50

R\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS10000) Range attribute), 55

raise\_exception() (msl.equipment.connection.Connection) attribute), 183

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS3000) Range attribute), 35

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS4000) Range attribute), 31

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS5000) Range attribute), 37

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS6000) Range attribute), 43

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS7000) Range attribute), 49

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS8000) Range attribute), 55

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS9000) Range attribute), 61

RAMP\_DOWN (msl.equipment.resources.picotech.picoscope.enums.PS10000) Range attribute), 67

RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS2000) Range attribute), 15



(msl.equipment.resources.picotech.picoscope.ps2000A.Picoscope2000A.Action\_msl.ConnectionMessageBased attribute), 83  
 read() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000A.Picoscope3000A attribute), 86  
 read() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10  
 RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps4000A.Picoscope4000A attribute), 89  
 read\_termination (msl.equipment.connection\_msl.ConnectionMessageBased method), 185  
 RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000A.Picoscope5000 attribute), 91  
 records() (msl.equipment.database.Database method), 191  
 RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000A.Picoscope5000A attribute), 92  
 read\_termination (msl.equipment.connection\_msl.ConnectionMessageBased method), 185  
 RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps6000A.Picoscope6000A attribute), 94  
 register (msl.equipment.record\_types.EquipmentRecord method), 126  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 22  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 31  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 43  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 38  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 56  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 52  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType attribute), 62  
 register\_message\_callback() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RAMPDOWN (msl.equipment.resources.picotech.picoscope.enums.PS2000WaveType attribute), 17  
 report\_error() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10  
 RAMPUP (msl.equipment.resources.picotech.picoscope.enums.PS2000WaveType attribute), 17  
 request\_backlash() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RANGES (msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelInfo attribute), 21  
 request\_button\_params() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RANGES (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelInfo attribute), 30  
 request\_button\_params() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RANGES (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelInfo attribute), 47  
 request\_button\_params() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RANGES (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelInfo attribute), 37  
 request\_button\_params() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RANGES (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelInfo attribute), 59  
 request\_button\_params() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 RANGES (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelInfo attribute), 54  
 request\_digital\_outputs() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 raw (msl.equipment.resources.picotech.picoscope.channel.PicoscopeChannel attribute), 13  
 request\_digital\_outputs() (msl.equipment.connection\_msl.ConnectionSerial method), 186  
 READ (msl.equipment.resources.picotech.picoscope.enums.PS4000AParameter attribute), 44  
 request\_digital\_outputs() (msl.equipment.connection\_msl.ConnectionSerial method), 186

method), 137

request\_hub\_bay()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

request\_io\_settings()  
(msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 125

request\_jog\_params()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 137

request\_led\_switches()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

request\_limit\_switch\_params()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 137

request\_mmi\_params()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

request\_move\_absolute\_position()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 137

request\_move\_relative\_distance()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_operating\_mode()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

request\_operating\_state()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

request\_position()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_potentiometer\_params()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_power\_params()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_settings()  
(msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 126

request\_settings()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_settings()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 150

request\_status()  
(msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 125

request\_status()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_status()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 151

request\_status\_bits()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_status\_bits()  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCubeSolenoid method), 151

request\_trigger\_config\_params()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 151

request\_trigger\_switches()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

request\_vel\_params()  
(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 138

RES\_12BIT (msl.equipment.resources.picotech.picoscope.enums.PicoscopeEnums attribute), 54

RES\_14BIT (msl.equipment.resources.picotech.picoscope.enums.PicoscopeEnums attribute), 54

RES\_15BIT (msl.equipment.resources.picotech.picoscope.enums.PicoscopeEnums attribute), 54

RES\_16BIT (msl.equipment.resources.picotech.picoscope.enums.PicoscopeEnums attribute), 54

RES\_8BIT (msl.equipment.resources.picotech.picoscope.enums.PicoscopeEnums attribute), 54

reserved (msl.equipment.resources.thorlabs.kinesis.structs.KLS\_MOTORS attribute), 169

reserved (msl.equipment.resources.thorlabs.kinesis.structs.KMOTORS attribute), 169

reserved (msl.equipment.resources.thorlabs.kinesis.structs.KMZ\_MOTORS attribute), 170

reserved (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_TRIPPER attribute), 174

reserved (msl.equipment.resources.thorlabs.kinesis.structs.KSC\_MOTORS attribute), 174

reserved (msl.equipment.resources.thorlabs.kinesis.structs.PS3 attribute), 100

reserved (msl.equipment.resources.thorlabs.kinesis.structs.PS5 attribute), 103

reserved (msl.equipment.resources.thorlabs.kinesis.structs.PS3 attribute), 100

reserved (msl.equipment.resources.picotech.picoscope.structs.PS5 attribute), 103

reserved (msl.equipment.resources.picotech.picoscope.structs.PS5 attribute), 103

reserved (msl.equipment.resources.thorlabs.kinesis.structs.FF\_IOS attribute), 167

reserved (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_5 attribute), 167

attribute), 159

reserved1 (msl.equipment.resources.thorlabs.kinesis.structs.PICOTEC\_SETTINGS), 188  
 attribute), 166

reserved2 (msl.equipment.resources.thorlabs.kinesis.structs.FRIO\_SETTINGS), 188  
 attribute), 167

reserved2 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StageAxisParameters), 162  
 attribute), 159

reserved3 (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_Direction), 121  
 attribute), 159

reserved4 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StageAxisParameters), 159  
 attribute), 159

reserved5 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StageAxisParameters), 160  
 attribute), 160

reserved6 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StageAxisParameters), 160  
 attribute), 160

reserved7 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StageAxisParameters), 160  
 attribute), 160

reserved8 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StageAxisParameters), 160  
 attribute), 160

reset\_stage\_to\_defaults()  
 (msl.equipment.resources.thorlabs.kinesis.instruments.integrated\_stepper\_motors.enums.PS3000Range), 138  
 method), 138

RESISTANCE\_100K  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 35  
 attribute), 35

RESISTANCE\_100R  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 35  
 attribute), 35

RESISTANCE\_10K  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 35  
 attribute), 35

RESISTANCE\_1K  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 35  
 attribute), 35

RESISTANCE\_1M  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 35  
 attribute), 35

RESISTANCE\_25K  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 36  
 attribute), 36

RESISTANCE\_50K  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 36  
 attribute), 36

RESISTANCE\_5K  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 36  
 attribute), 36

RESISTANCES (msl.equipment.resources.picotech.picoscope.enums.PS4000Channels), 47  
 attribute), 47

RESISTANCES (msl.equipment.resources.picotech.picoscope.enums.PS4000Channels), 47  
 attribute), 37

resource\_manager()  
 (msl.equipment.connection\_pyvisa.ConnectionPyVISA), 188  
 resource\_pyclass()  
 (msl.equipment.connection\_pyvisa.ConnectionPyVISA), 188  
 static method), 188

rising (msl.equipment.resources.picotech.picoscope.enums.PS2000Direction), 33  
 attribute), 33

RISING (msl.equipment.resources.picotech.picoscope.enums.PS3000Direction), 33  
 attribute), 33

RISING (msl.equipment.resources.picotech.picoscope.enums.PS3000Direction), 33  
 attribute), 27

RISING (msl.equipment.resources.picotech.picoscope.enums.PS3000Direction), 33  
 attribute), 45

RISING (msl.equipment.resources.picotech.picoscope.enums.PS4000Direction), 39  
 attribute), 39

RISING (msl.equipment.resources.picotech.picoscope.enums.PS4000Direction), 39  
 attribute), 57

RISING (msl.equipment.resources.picotech.picoscope.enums.PS5000Direction), 62  
 attribute), 62

RISING (msl.equipment.resources.picotech.picoscope.enums.PS6000Direction), 63  
 attribute), 63

RISING\_LOWER



SAMPLE\_PROPERTIES\_POST\_TRIGGER\_SAMPLES (msl.equipment.resources.picotech.picoscope.enums.PS1000A.PicoStringValue attribute), 120  
 (msl.equipment.resources.picotech.picoscope.enums.PS1000A.PicoStringValue attribute), 22

SAMPLE\_PROPERTIES\_PRE\_TRIGGER\_SAMPLES (msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoStringValue attribute), 48  
 SCOPE\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoStringValue attribute), 45

SAMPLE\_PROPERTIES\_RESOLUTION (msl.equipment.resources.picotech.picoscope.enums.PS1000A.PicoStringValue attribute), 49  
 attribute), 38

SAMPLE\_PROPERTIES\_TIMEBASE (msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoStringValue attribute), 49  
 SCOPE\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoStringValue attribute), 52

samplesPerRevolution (msl.equipment.resources.thorlabs.kinesis.structs.SCOPENTRIG.Parameters attribute), 163  
 attribute), 62

save() (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C.Connection\_msl.ConnectionSDK method), 177  
 attribute), 185

save\_setup() (msl.equipment.resources.bentham.benhw32.Bentham32.SDK.BenthamEquipment.constants.MSLInterface attribute), 10  
 attribute), 189

save\_streaming\_data() (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000 method), 84  
 sdk\_path (msl.equipment.connection\_msl.ConnectionSDK attribute), 183

SC\_Active (msl.equipment.resources.thorlabs.kinesis.enums.SCOperatingStates attribute), 120  
 section (msl.equipment.record\_types.EquipmentRecord attribute), 185

SC\_Auto (msl.equipment.resources.thorlabs.kinesis.enums.SCOperatingModes attribute), 120  
 segmentIndex (msl.equipment.resources.picotech.picoscope.structs.SC\_CycleParameters attribute), 100

SC\_CycleParameters (class in msl.equipment.resources.thorlabs.kinesis.structs) segmentIndex (msl.equipment.resources.picotech.picoscope.structs attribute), 173  
 attribute), 103

SC\_Inactive (msl.equipment.resources.thorlabs.kinesis.enums.SCOperatingStates attribute), 120  
 select\_wavelength() (msl.equipment.resources.bentham.benhw32.Bentham32 attribute), 10

SC\_Manual (msl.equipment.resources.thorlabs.kinesis.enums.SCOperatingModes attribute), 120  
 select\_wavelength() (msl.equipment.resources.bentham.benhw64.Bentham attribute), 11

SC\_OperatingModes (class in msl.equipment.resources.thorlabs.kinesis.enums), method), 11  
 120  
 selectedRange (msl.equipment.resources.thorlabs.kinesis.structs.NT attribute), 164

SC\_OperatingStates (class in msl.equipment.resources.thorlabs.kinesis.enums) send() (msl.equipment.connection\_msl.ConnectionMessageBased attribute), 120  
 method), 186

SC\_Single (msl.equipment.resources.thorlabs.kinesis.enums.SCOperatingModes attribute), 120  
 method), 10

SC\_SolenoidClosed (msl.equipment.resources.thorlabs.kinesis.enums.SC(SolenoidState resources.picotech.picoscope.enums.PS4000A.PicoStringValue attribute), 121  
 attribute), 46

SC\_SolenoidOpen (msl.equipment.resources.thorlabs.kinesis.enums.SC(SolenoidState resources.thorlabs.fw102c), attribute), 121  
 175

SC\_SolenoidStates (class in serial (msl.equipment.connection\_msl.ConnectionSerial msl.equipment.resources.thorlabs.kinesis.enums), attribute), 187  
 120  
 serial (msl.equipment.record\_types.ConnectionRecord attribute), 194

SC\_Triggered (msl.equipment.resources.thorlabs.kinesis.enums.SCOperatingModes attribute), 120

serial (msl.equipment.record\_types.EquipmentRecord.set\_cycle\_params()  
 attribute), 193 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid  
 SERIAL\_NUMBER\_BUFFER\_SIZE (msl.equipment.resources.thorlabs.kinesis.sections.central\_method\_control  
 attribute), 156 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid  
 serialNo (msl.equipment.resources.thorlabs.kinesis.structs.TLHDeviceInfo  
 attribute), 157 set\_data\_buffer() (msl.equipment.resources.picotech.picoscope.pico  
 serialNumber (msl.equipment.resources.thorlabs.kinesis.structs.MIOI)StageAxisParameters  
 attribute), 160 set\_data\_buffer\_bulk()  
 serialNumber (msl.equipment.resources.thorlabs.kinesis.structs.TLSEquipmentInformation)picotech.picoscope.picoscope\_a  
 attribute), 158 method), 76  
 set() (msl.equipment.resources.bentham.benhw32.Bentham.set\_data\_buffer\_with\_mode()  
 method), 10 (msl.equipment.resources.picotech.picoscope.ps4000.Pico  
 set() (msl.equipment.resources.bentham.benhw64.Bentham method), 88  
 method), 11 set\_data\_buffers() (msl.equipment.resources.picotech.picoscope.pico  
 set\_acceleration() (msl.equipment.resources.thorlabs.fw102c.Firmware)FW102C  
 method), 177 set\_data\_buffers\_bulk()  
 set\_adv\_trigger\_channel\_conditions() (msl.equipment.resources.picotech.picoscope.ps6000.Pico  
 (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k)PicoScope2k3k  
 method), 71 set\_data\_buffers\_with\_mode()  
 set\_adv\_trigger\_channel\_directions() (msl.equipment.resources.picotech.picoscope.ps4000.Pico  
 (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k)PicoScope2k3k  
 method), 70 set\_device\_resolution()  
 set\_adv\_trigger\_channel\_properties() (msl.equipment.resources.picotech.picoscope.ps5000a.Pico  
 (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k)PicoScope2k3k  
 method), 71 set\_digital\_analog\_trigger\_operand()  
 set\_adv\_trigger\_delay() (msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
 (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k)PicoScope2k3k  
 method), 70 set\_digital\_outputs()  
 set\_backlash() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors.kcube\_solenoid  
 method), 139 method), 151  
 set\_bandwidth\_filter() set\_digital\_port() (msl.equipment.resources.picotech.picoscope.pico  
 (msl.equipment.resources.picotech.picoscope.picoscope\_api)PicoScopeApi  
 method), 76 set\_direction() (msl.equipment.resources.thorlabs.kinesis.integrated  
 set\_bow\_index() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors  
 method), 139 set\_ets() (msl.equipment.resources.picotech.picoscope.picoscope\_2  
 set\_button\_params() method), 71  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors.picoscope.picoscope\_a  
 method), 139 method), 77  
 set\_button\_params\_block() set\_ets\_time\_buffer()  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors.picoscope.picoscope\_a  
 method), 139 method), 77  
 set\_bw\_filter() (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000  
 method), 88 (msl.equipment.resources.picotech.picoscope.picoscope\_a  
 set\_calibration\_file() method), 77  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors  
 method), 139 (msl.equipment.resources.picotech.picoscope.ps4000.Pico  
 set\_channel() (msl.equipment.resources.picotech.picoscope.picoscope\_2k3k)PicoScope  
 method), 68 set\_external\_clock()  
 set\_channel\_led() (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000)picotech.picoscope.ps6000.Pico  
 method), 90 method), 94

[set\\_frequency\\_counter\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [PicoScopeApi](#) method), 77 [set\\_motor\\_params\(\)](#)  
[set\\_homing\\_params\\_block\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 140 [set\\_motor\\_params\\_ext\(\)](#)  
[set\\_homing\\_velocity\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 140 [set\\_motor\\_travel\\_limits\(\)](#)  
[set\\_io\\_settings\(\)](#) (msl.equipment.resources.thorlabs.kinesis.filter\_wheel) [FilterWheel](#) method), 125 [set\\_no\\_of\\_captures\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 143  
[set\\_jog\\_mode\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 140 [set\\_operating\\_mode\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 143  
[set\\_jog\\_params\\_block\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 140 [set\\_operating\\_state\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 143  
[set\\_jog\\_step\\_size\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 141 [set\\_position\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 143  
[set\\_jog\\_vel\\_params\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 141 [set\\_potentiometer\\_params\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 143  
[set\\_led\(\)](#) (msl.equipment.resources.picotech.picoscope.ps2000) [PicoScope2000](#) method), 81 [set\\_no\\_of\\_captures\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 151 [set\\_operating\\_state\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 151  
[set\\_led\\_switches\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 141 [set\\_operating\\_mode\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 151  
[set\\_led\\_switches\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 151 [set\\_operating\\_state\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 151  
[set\\_light\(\)](#) (msl.equipment.resources.picotech.picoscope.ps2000) [PicoScope2000](#) method), 81 [set\\_position\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 152  
[set\\_limit\\_switch\\_params\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 141 [set\\_position\\_count\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 152  
[set\\_limit\\_switch\\_params\\_block\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 142 [set\\_position\\_counter\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 152  
[set\\_limits\\_software\\_approach\\_policy\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 142 [set\\_potentiometer\\_params\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 143  
[set\\_max\\_velocity\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 144 [set\\_potentiometer\\_params\\_block\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 144  
[set\\_min\\_velocity\(\)](#) (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors) [IntegratedStepperMotors](#) method), 144 [set\\_position\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 152  
[set\\_mmi\\_params\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 152 [set\\_probe\(\)](#) (msl.equipment.resources.picotech.picoscope.ps4000) [PicoScope4000](#) method), 89  
[set\\_mmi\\_params\\_ext\(\)](#) (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) [KCubeSolenoid](#) method), 89

set\_pulse\_width\_digital\_port\_properties() (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000) method), 68  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000) method), 86  
 set\_pulse\_width\_qualifier() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 69  
 set\_pulse\_width\_qualifier\_conditions() (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A) method), 90  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A) method), 71  
 set\_pulse\_width\_qualifier\_properties() (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A) method), 90  
 set\_pulse\_width\_qualifier\_v2() (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000A) method), 86  
 set\_sensor\_mode() (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C) method), 178  
 set\_sig\_gen\_arbitrary() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 77  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A) method), 90  
 set\_sig\_gen\_arbitrary() (msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000) method), 81  
 set\_sig\_gen\_built\_in() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 78  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) method), 152  
 set\_sig\_gen\_built\_in() (msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000) method), 82  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid) method), 153  
 set\_sig\_gen\_built\_in\_v2() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 78  
 set\_sig\_gen\_properties\_arbitrary() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 79  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000A) method), 83  
 set\_sig\_gen\_properties\_built\_in() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 79  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000A) method), 86  
 set\_sigen() (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000) method), 84  
 (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C) method), 178  
 set\_simple\_trigger() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScopeApi) method), 79  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_steps) method), 145  
 set\_speed\_mode() (msl.equipment.resources.thorlabs.fw102c.FilterWheel102C) method), 178  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_steps) method), 145  
 set\_stage\_axis\_limits() (msl.equipment.resources.thorlabs.kinesis.integrated\_steps) method), 145  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_steps) method), 145  
 set\_timebase() (msl.equipment.resources.picotech.picoscope.picoscope\_api.PicoScope



(msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope40000 (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope40000 method), 94 SIGNAL\_GENERATOR\_AWG\_WAVEFORM\_SIZE (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope40000 attribute), 49

settledError (msl.equipment.resources.thorlabs.kinesis.structs.MOTEDriversTestResultsSet.PotentialPicoscope.enums.PS4000a.PicoScope40000 attribute), 161 attribute), 49

settleSamples (msl.equipment.resources.thorlabs.kinesis.structs.MOTEDriversTestResultsSet.PotentialPicoscope.enums.PS4000a.PicoScope40000 attribute), 164 SIGNAL\_GENERATOR\_BUILT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49

SEVEN (msl.equipment.constants.DataBits attribute), 190 SIGNAL\_GENERATOR\_BUILT\_IN\_DWELL\_TIME (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49

SHORT\_PRESS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 16 (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49

SHOT\_SWEEP\_TRIGGER\_CONTINUOUS\_RUN SIGNAL\_GENERATOR\_BUILT\_IN\_INCREMENT (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000a (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 83 attribute), 49

SHOT\_SWEEP\_TRIGGER\_CONTINUOUS\_RUN SIGNAL\_GENERATOR\_BUILT\_IN\_START\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000a (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 86 attribute), 49

SHOT\_SWEEP\_TRIGGER\_CONTINUOUS\_RUN SIGNAL\_GENERATOR\_BUILT\_IN\_STOP\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000a (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 92 attribute), 49

sig\_gen\_arbitrary\_min\_max\_values() SIGNAL\_GENERATOR\_BUILT\_IN\_WAVE\_TYPE (msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScopeApi (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 method), 79 attribute), 49

sig\_gen\_frequency\_to\_phase() SIGNAL\_GENERATOR\_EXT\_IN\_THRESHOLD (msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScopeApi (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 method), 79 attribute), 49

sig\_gen\_software\_control() SIGNAL\_GENERATOR\_OFFSET\_VOLTAGE (msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScopeApi (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 method), 80 attribute), 49

SIGNAL\_GENERATOR SIGNAL\_GENERATOR\_OPERATION (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

SIGNAL\_GENERATOR\_ARRAY\_OF\_AWG\_WAVEFORMS SIGNAL\_GENERATOR\_PK\_TO\_PK (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

SIGNAL\_GENERATOR\_AWG SIGNAL\_GENERATOR\_SETTINGS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 44

SIGNAL\_GENERATOR\_AWG\_DELTA\_PHASE\_INCREMENT SIGNAL\_GENERATOR\_SHOTS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

SIGNAL\_GENERATOR\_AWG\_DWELL\_COUNT SIGNAL\_GENERATOR\_SWEEP\_TYPE (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

SIGNAL\_GENERATOR\_AWG\_INDEX\_MODE SIGNAL\_GENERATOR\_SWEEPS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

SIGNAL\_GENERATOR\_AWG\_START\_DELTA\_PHASE SIGNAL\_GENERATOR\_TRIGGER\_SOURCE (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

SIGNAL\_GENERATOR\_AWG\_STOP\_DELTA\_PHASE SIGNAL\_GENERATOR\_TRIGGER\_TYPE (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope40000 attribute), 49 attribute), 49

attribute), 49

SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 21

SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 17

SINE (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 31

SINE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 43

SINE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 38

SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 52

SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 62

SINE (msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType attribute), 94

SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 17

SINE (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 25

SINE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 38

SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 52

SINE (msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType attribute), 94

SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000A attribute), 83

SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000A attribute), 86

SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A attribute), 89

SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000A attribute), 92

SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope6000A attribute), 93

SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 32

SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 35

SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 45

SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 57

SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType attribute), 66

SIX (msl.equipment.constants.DataBits attribute),

SIX (msl.equipment.resources.thorlabs.fw102c.FilterCount attribute),

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 21

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 17

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 31

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 25

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 43

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 38

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 56

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 52

attribute), 56

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_SlowMode attribute), 51

SLOW (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_SlowMode attribute), 61

SLOW (msl.equipment.resources.thorlabs.fw102c.SpeedMode attribute), 89

SLOW (msl.equipment.resources.thorlabs.fw102c.SpeedMode attribute), 175

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_SigGenTrigSource attribute), 22

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_SigGenTrigSource attribute), 32

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_SigGenTrigSource attribute), 45

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_SigGenTrigSource attribute), 38

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_SigGenTrigSource attribute), 57

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_SigGenTrigSource attribute), 52

SOFT\_TRIG (msl.equipment.resources.picotech.picoscope.enums.PS6000A\_SigGenTrigSource attribute), 62

softLimitMode (msl.equipment.resources.thorlabs.kinesis\_stoppingModes.LimitSwitchParameters attribute), 162

source (msl.equipment.resources.picotech.picoscope.structs.PS4000A\_Sharding attribute), 102

SPACE (msl.equipment.constants.Parity attribute), 189

SpeedMode (class in msl.equipment.resources.thorlabs.fw102c), 175

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_WaveType attribute), 21

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_WaveType attribute), 17

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_WaveType attribute), 31

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_WaveTypes attribute), 25

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_WaveType attribute), 43

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_WaveType attribute), 38

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_WaveType attribute), 56

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_WaveType attribute), 52

SQUARE (msl.equipment.resources.picotech.picoscope.enums.PS6000A\_WaveType attribute), 61

SQUARE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_SigGenTrigSource attribute), 83

SQUARE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_SigGenTrigSource attribute), 86

SQUARE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_SigGenTrigSource attribute), 89

SQUARE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_SigGenTrigSource attribute), 91

SQUARE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_SigGenTrigSource attribute), 94

SQUARE\_MAX\_FREQUENCY (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_State attribute), 160

startLog() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 10

startPolling() (msl.equipment.resources.thorlabs.kinesis.filter\_flipping method), 126

startStoppingModes() (msl.equipment.resources.thorlabs.kinesis.integrated method), 146

start() (msl.equipment.resources.thorlabs.kinesis.kcube\_sol method), 153

state (msl.equipment.resources.picotech.picoscope.structs.PS4000A attribute), 101

state (msl.equipment.resources.picotech.picoscope.structs.PS4000A attribute), 102

status (msl.equipment.resources.picotech.picoscope.structs.PS3000A attribute), 102

status (msl.equipment.resources.picotech.picoscope.structs.PS5000A attribute), 102

stepSize (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_Job attribute), 102

stop() (msl.equipment.resources.picotech.picoscope.picoscope.Pico attribute), 102

stop\_bits (msl.equipment.connection\_msl.ConnectionSerial attribute), 102

stop\_immediate() (msl.equipment.resources.thorlabs.kinesis.integrated method), 102

stop\_log() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 102

stop\_polling() (msl.equipment.resources.thorlabs.kinesis.filter\_flipping method), 102

stop\_polling() (msl.equipment.resources.thorlabs.kinesis.integrated method), 102

stop\_polling() (msl.equipment.resources.thorlabs.kinesis.kcube\_sol method), 102

stop() (msl.equipment.resources.thorlabs.kinesis.integrated method), 146

StopBits (class in msl.equipment.constants), 189

stopMode (msl.equipment.resources.thorlabs.kinesis.structs.MOTION\_CONTROL (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 158

StrainGauge (msl.equipment.resources.thorlabs.kinesis.structs.MOTION\_CONTROL (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 115

streaming\_ns\_get\_interval\_stateless() (msl.equipment.resources.picotech.picoscope.enums.PS4000\_DATA\_SCOPE\_3000 method), 84

structs() (msl.equipment.resources.utils.CHeader attribute), 179

sum (msl.equipment.resources.thorlabs.kinesis.structs.QD\_READING (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 172

SWOnly (msl.equipment.resources.thorlabs.kinesis.enums.TC\_CUBE\_PC\_CARD\_IOCTL\_MODE (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 115

## T

TC\_ActualTemperature (msl.equipment.resources.thorlabs.kinesis.enums.TC\_ActualTemperatureModes attribute), 123

TC\_Current (msl.equipment.resources.thorlabs.kinesis.enums.TC\_DisplayModes attribute), 123

TC\_DisplayModes (class in TCube\_Strain\_Gauge (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 123

TC\_LoopParameters (class in TCube\_TEC (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 174

TC\_SensorTypes (class in (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 123

TC\_TargetTemperature (msl.equipment.resources.thorlabs.kinesis.enums.TC\_DisplayModes attribute), 123

TC\_TempDifference (msl.equipment.resources.thorlabs.kinesis.enums.TC\_DisplayModes attribute), 123

TC\_TH200kOhm (msl.equipment.resources.thorlabs.kinesis.enums.TC\_SensorTypes attribute), 123

TC\_TH20kOhm (msl.equipment.resources.thorlabs.kinesis.enums.TC\_SensorTypes attribute), 123

TC\_Transducer (msl.equipment.resources.thorlabs.kinesis.enums.TC\_SensorTypes attribute), 123

TCPIP (msl.equipment.constants.MSLInterface attribute), 189

TCPIP\_ASRL (msl.equipment.constants.MSLInterface attribute), 189

TCPIP\_GPIB (msl.equipment.constants.MSLInterface attribute), 189

TCube\_Brushless\_Motor (msl.equipment.resources.thorlabs.kinesis.structs.MOTION\_CONTROL (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TCube\_DC\_Servo (msl.equipment.resources.thorlabs.kinesis.structs.MOTION\_CONTROL (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TCube\_LaserSource (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TCube\_PC\_Card\_IOCTL\_Mode (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TCube\_Quad (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TCube\_Solenoid (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TCube\_Stepper\_Motor (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 155

TEMPERATURE\_SENSOR (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 36

TEMPERATURE\_UPTO\_100 (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 36

TEMPERATURE\_UPTO\_130 (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 36

TEMPERATURE\_UPTO\_40 (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 36

TEMPERATURE\_SENSOR\_TYPES (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 36

TEMPERATURES (msl.equipment.resources.picotech.picoscope.enums.PS4000\_SENSOR\_TYPES attribute), 36

THEN (msl.equipment.resources.picotech.picoscope.enums.PS2000\_THRESHOLD\_DEFLECTION attribute), 19

thresholdDeflection (msl.equipment.resources.thorlabs.kinesis.structs.MOTION\_CONTROL (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 168

brushlessMotorMotionControl (msl.equipment.resources.picotech.picoscope.structs.MOTION\_CONTROL (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 96

thresholdLower (msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerChannelProperties attribute), 99

thresholdLower (msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProperties attribute), 102

thresholdLower (msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerChannelProperties attribute), 101

thresholdLower (msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerChannelProperties attribute), 104

thresholdLower (msl.equipment.resources.picotech.picoscope.structs.PS6000TriggerChannelProperties attribute), 106

thresholdLowerHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerChannelProperties attribute), 96

thresholdLowerHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerChannelProperties attribute), 99

thresholdLowerHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProperties attribute), 102

thresholdLowerHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerChannelProperties attribute), 101

thresholdLowerHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerChannelProperties attribute), 104

thresholdMajor (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 94

thresholdMajor (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 97

thresholdMajor (msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerChannelProperties attribute), 103

thresholdMinor (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 95

thresholdMinor (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 97

thresholdMinor (msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerChannelProperties attribute), 103

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerChannelProperties attribute), 96

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 95

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerChannelProperties attribute), 100

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 97

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProperties attribute), 102

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerChannelProperties attribute), 101

thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerChannelProperties attribute), 105

thresholdUpper (msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerChannelProperties attribute), 105

thresholdUpper (msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerChannelProperties attribute), 106

thresholdUpperHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProperties attribute), 97

thresholdUpperHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerChannelProperties attribute), 100

thresholdUpperHysteresis (msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerChannelProperties attribute), 102

thresholdUpperHysteresis (msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.structs),

TIM\_ButtonsMode (class in msl.equipment.resources.thorlabs.kinesis.enums), 121

TIM\_Direction (class in msl.equipment.resources.thorlabs.kinesis.enums), 121

TIM\_Direction (class in msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.structs),

TIM\_JogMode (class in msl.equipment.resources.thorlabs.kinesis.enums), 121

TIM\_JogParams (class in msl.equipment.resources.thorlabs.kinesis.structs), msl.equipment.resources.thorlabs.kinesis.structs),

174 TRIANGLE (msl.equipment.resources.picotech.picoscope.enums.F  
TIM\_Status (class in attribute), 25  
msl.equipment.resources.thorlabs.kinesis.structs),  
174 TRIANGLE (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 43  
TIME (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 24  
TIME (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 34  
TIME (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 40  
time (msl.equipment.resources.thorlabs.kinesis.structs),  
attribute), 161  
time\_since\_last\_msg\_received() TRIANGLE\_MAX\_FREQUENCY  
(msl.equipment.resources.thorlabs.kinesis.filter\_flipped),  
method), 126 attribute), 83  
time\_since\_last\_msg\_received() TRIANGLE\_MAX\_FREQUENCY  
(msl.equipment.resources.thorlabs.kinesis.integrated),  
method), 146 attribute), 86  
time\_since\_last\_msg\_received() TRIANGLE\_MAX\_FREQUENCY  
(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid),  
method), 153 attribute), 89  
timeout (msl.equipment.resources.thorlabs.kinesis.structs),  
attribute), 168  
timeStampCounter TRIANGLE\_MAX\_FREQUENCY  
(msl.equipment.resources.picotech.picoscope.ps2000a),  
attribute), 100  
timeUnits (msl.equipment.resources.picotech.picoscope.structs),  
attribute), 100  
timeUnits (msl.equipment.resources.picotech.picoscope.structs),  
attribute), 103  
TLI\_DeviceInfo (class in trig1DiffThreshold  
msl.equipment.resources.thorlabs.kinesis.structs),  
157 attribute), 172  
TLI\_HardwareInformation (class in trig1Mode (msl.equipment.resources.thorlabs.kinesis.structs.QD\_K  
msl.equipment.resources.thorlabs.kinesis.structs),  
157 attribute), 173  
to\_version() (msl.equipment.resources.thorlabs.kinesis.motion),  
static method), 156  
TPZ\_IOSettings (class in attribute), 173  
msl.equipment.resources.thorlabs.kinesis.structs),  
170 attribute), 173  
trace() (msl.equipment.resources.bentham.benhw32),  
method), 10  
transitTime (msl.equipment.resources.thorlabs.kinesis.structs),  
attribute), 167  
TRIANGLE (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 21  
TRIANGLE (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 17  
TRIANGLE (msl.equipment.resources.picotech.picoscope.enums.F  
attribute), 31

attribute), 173	TRIGGER_DIRECTION
TRIGGER (msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue (msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue attribute), 47	attribute), 48
Trigger1Mode (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	TRIGGER_DIRECTION_CHANNEL
attribute), 169	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
Trigger1Mode (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	attribute), 48
attribute), 170	TRIGGER_DIRECTION_DIRECTION
Trigger1Mode (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 174	attribute), 48
Trigger1Polarity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	TRIGGER_NO_OF_BLOCK_CONDITIONS
attribute), 169	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
Trigger1Polarity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	attribute), 48
attribute), 170	TRIGGER_NO_OF_CONDITIONS
Trigger1Polarity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 174	attribute), 48
Trigger2Mode (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	TRIGGER_NO_OF_TRIGGER_CONDITIONS
attribute), 169	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
Trigger2Mode (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	attribute), 48
attribute), 170	TRIGGER_PROPERTIES
Trigger2Mode (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 174	attribute), 47
Trigger2Polarity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	TRIGGER_PROPERTIES_CHANNEL
attribute), 169	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
Trigger2Polarity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	attribute), 48
attribute), 171	TRIGGER_PROPERTIES_THRESHOLD_LOWER
Trigger2Polarity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTTriggerConfig	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 174	attribute), 48
TRIGGER_ARRAY_OF_BLOCK_CONDITIONS	TRIGGER_PROPERTIES_THRESHOLD_LOWER_HYSTERESIS
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 48	attribute), 48
TRIGGER_AUTO_TRIGGER_MILLISECONDS	TRIGGER_PROPERTIES_THRESHOLD_MODE
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 47	attribute), 48
TRIGGER_AUXIO_OUTPUT_ENABLED	TRIGGER_PROPERTIES_THRESHOLD_UPPER
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 47	attribute), 47
TRIGGER_CONDITION_SOURCE	TRIGGER_PROPERTIES_THRESHOLD_UPPER_HYSTERESIS
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
attribute), 48	attribute), 47
TRIGGER_CONDITION_STATE	TRIGGER_RAW (msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	trigger_within_pre_trigger_samples()
attribute), 48	(msl.equipment.resources.picotech.picoscope.picoscope_a
TRIGGER_CONDITIONS	(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	triggerIndex (msl.equipment.resources.picotech.picoscope.structs.P
attribute), 48	attribute), 103
TRIGGER_DELAY	TriggerIndex (msl.equipment.resources.thorlabs.kinesis.structs.KMOT
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	attribute), 169
attribute), 48	(msl.equipment.resources.thorlabs.kinesis.structs.KMOT
TRIGGER_DELAY_MS	TriggerIndex (msl.equipment.resources.thorlabs.kinesis.structs.KMOT
(msl.equipment.resources.picotech.picoscope.enums.PS4000APIProtoStringValue	(msl.equipment.resources.thorlabs.kinesis.structs.KMOT
attribute), 48	(msl.equipment.resources.thorlabs.kinesis.structs.KMOT

attribute), 169

TriggerMode (class in TSG\_HubChannel2 (msl.equipment.resources.thorlabs.fw102c), attribute), 175

TriggerPulseCountFwd (class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TriggerParams), attribute), 169

TriggerPulseCountRev (class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TriggerParams), attribute), 169

TriggerPulseWidth (class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TriggerParams), attribute), 169

TriggerStartPositionFwd (class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TriggerParams), attribute), 169

TriggerStartPositionRev (class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TriggerParams), attribute), 169

triggerTime (msl.equipment.resources.picotech.picoscope.structs.PS1000ATTriggerInfo attribute), 100

triggerTime (msl.equipment.resources.picotech.picoscope.structs.PS5000ATTriggerInfo attribute), 104

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS2000ATTriggerState attribute), 24

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS2000ATTriggerState attribute), 18

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS3000ATTriggerState attribute), 33

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS3500ATTriggerState attribute), 27

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS4000ATTriggerState attribute), 46

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS4000ATTriggerState attribute), 39

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS5000ATTriggerState attribute), 58

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS5000ATTriggerState attribute), 53

TRUE (msl.equipment.resources.picotech.picoscope.enums.PS6000ATTriggerState attribute), 63

TSG\_Display\_Modes (class in UP (msl.equipment.resources.picotech.picoscope.enums.PS2000ASW\_TSG\_Display\_Modes), attribute), 21

TSG\_Force (msl.equipment.resources.thorlabs.kinesis.enums.TSG\_Display\_Modes attribute), 123

TSG\_Hub\_Analogue\_Modes (class in UP (msl.equipment.resources.thorlabs.kinesis.enums.TSG\_Hub\_Analogue\_Modes), attribute), 122

TSG\_HubChannel1 (class in UP (msl.equipment.resources.thorlabs.kinesis.enums.TSG\_HubChannel1), attribute), 122

attribute), 122

(msl.equipment.resources.thorlabs.kinesis.enums.TSG\_HubChannel2 attribute), 122

(class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs), attribute), 174

(class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.structs), attribute), 174

(msl.equipment.resources.thorlabs.kinesis.enums.TSG\_Position attribute), 174

(msl.equipment.resources.thorlabs.kinesis.enums.TSG\_Undefined attribute), 174

(msl.equipment.resources.thorlabs.kinesis.enums.TSG\_KMOT\_TriggerParams attribute), 122

(msl.equipment.resources.thorlabs.kinesis.enums.TSG\_KMOT\_TriggerParams attribute), 122

TWELVE (msl.equipment.resources.thorlabs.fw102c.FilterCount attribute), 174

TWO (msl.equipment.constants.StopBits attribute), 189

type (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Hardware attribute), 158

type (msl.equipment.resources.thorlabs.kinesis.structs.TLI\_Device attribute), 157

under OP2000ATTriggerState (msl.equipment.resources.thorlabs.kinesis.structs.NT\_TIA attribute), 16

UNIT\_INFO (msl.equipment.resources.picotech.picoscope.enums.PS3000ATTriggerState attribute), 44

UnitType (class in TSG\_IOSettings (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper.enums.TSG\_IOSettings), attribute), 127

UNKNOWN (msl.equipment.constants.Backend attribute), 188

UNKNOWN (msl.equipment.resources.picotech.picoscope.enums.PS4000ATTriggerState attribute), 36

unused PS5000ATTriggerState (msl.equipment.resources.thorlabs.kinesis.structs.KSC\_MMIO attribute), 174

unused PS5000ATTriggerState (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_Bus attribute), 168

unused PS6000ATTriggerState (msl.equipment.resources.thorlabs.kinesis.structs.NT\_IOSettings attribute), 165

UP (msl.equipment.resources.picotech.picoscope.enums.PS4000ASW attribute), 43

UP (msl.equipment.resources.picotech.picoscope.enums.PS4000SW attribute), 37





wait\_for\_message() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors method), 146

wait\_for\_message() (msl.equipment.resources.thorlabs.kinesis.kinesis\_structs.QD\_Position attribute), 173

wait\_until\_ready() (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope method), 69

wavelength (msl.equipment.resources.bentham.benhw64.Bentham method), 187

wavelength (msl.equipment.resources.bentham.benhw64.Bentham attribute), 11

wDigOPs (msl.equipment.resources.thorlabs.kinesis.structs.QD\_KRA\_DigitalIO attribute), 173

WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 44

WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType attribute), 38

WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 57

WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS10000WaveType attribute), 52

WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS2000AExtraOperations attribute), 22

WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS3000AExtraOperations attribute), 32

WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations attribute), 40

WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000OperationTypes attribute), 38

WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS5000AExtraOperations attribute), 54

WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS6000ExtraOperations attribute), 62

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_ThresholdMode attribute), 23

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS2000ThresholdMode attribute), 17

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_ThresholdMode attribute), 32

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdMode attribute), 27

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdMode attribute), 45

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdMode attribute), 39

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdMode attribute), 57

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdMode attribute), 53

WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdMode attribute), 63

wReserved (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 160

wReserved (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 173

WRITE (msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType attribute), 44

write() (msl.equipment.connection\_msl.ConnectionMessageBased method), 86

write() (msl.equipment.connection\_msl.ConnectionSerial method), 187

write\_termination (msl.equipment.connection\_msl.ConnectionMessageBased method), 187

X

x (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 173

xFeedbackSignedGain (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 172

XML (msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType attribute), 44

Y

y (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 172

zero\_calibration() (msl.equipment.resources.bentham.benhw32.Bentham method), 187

zero\_calibration() (msl.equipment.resources.bentham.benhw64.Bentham method), 187

Z