

---

# **MSL-Equipment Documentation**

***Release 0.1.0***

**Measurement Standards Laboratory of New Zealand**

**Jun 17, 2023**



## CONTENTS

|          |                            |            |
|----------|----------------------------|------------|
| <b>1</b> | <b>Contents</b>            | <b>3</b>   |
| <b>2</b> | <b>Index</b>               | <b>597</b> |
|          | <b>Python Module Index</b> | <b>599</b> |
|          | <b>Index</b>               | <b>601</b> |



The purpose of MSL-Equipment is to manage information about equipment that are required to perform a measurement and to connect to equipment that support computer control. Three items are used to achieve this purpose

1. *Configuration File*
2. *Equipment-Register Database*
3. *Connections Database*

The following example uses a [configuration file](#) that specifies an [equipment-register database](#) (which contains information about all of the equipment that is available), a [connections database](#) (which contains information on how to connect to equipment that support computer control), and specifies a digital multimeter to use for a measurement (which is defined as an `<equipment>` XML element).

Loading the [configuration file](#) using the [Config](#) class is the main entry point

```
>>> from msl.equipment import Config
>>> cfg = Config('config.xml')
```

and loading the [Databases](#) is done via the `database()` method

```
>>> db = cfg.database()
```

You can access XML elements, attributes and values in the [configuration file](#)

```
>>> cfg.find('max_voltage')
<Element 'max_voltage' at ...>
>>> cfg.attrib('max_voltage')
{'unit': 'V'}
>>> cfg.value('max_voltage')
3.3
```

and all of the [EquipmentRecord](#)'s that are contained within the [Equipment-Register Database](#)<sup>1</sup>

```
>>> for record in db.records():
...     print(record)
...
EquipmentRecord<Fluke|8506A|cecf2e0a>
EquipmentRecord<Oriel|66087|169b71e9>
EquipmentRecord<Kepco|JQE|baee83d4>
EquipmentRecord<Hewlett Packard|34401A|15ab8c6c>
EquipmentRecord<Arlunya|Milli Gauss|890a4c02>
EquipmentRecord<Toledo|1000|444213b7>
EquipmentRecord<Stanford Research Systems|SR850 DSP|cec817f5>
EquipmentRecord<HP|3478A|bd92c887>
```

To select the records that are from Hewlett Packard<sup>1</sup>, specify the *manufacturer* as a search criteria (supports [regex](#), so both *HP* and *Hewlett Packard* will match)

<sup>1</sup> Companies that sell equipment that is used for scientific research are identified in this guide in order to illustrate how to adequately use MSL-Equipment in your own application. Such identification is not intended to imply recommendation or endorsement by the Measurement Standards Laboratory of New Zealand, nor is it intended to imply that the companies identified are necessarily the best for the purpose.

```
>>> for record in db.records(manufacturer='H.*P'):  
...     print(record)  
...  
EquipmentRecord<Hewlett Packard|34401A|15ab8c6c>  
EquipmentRecord<HP|3478A|bd92c887>
```

Get the *ConnectionRecord*'s of the equipment that can be computer controlled

```
>>> for conn in db.connections():  
...     print(conn)  
...  
ConnectionRecord<Fluke|8506A|cecf2e0a>  
ConnectionRecord<Hewlett Packard|34401A|15ab8c6c>  
ConnectionRecord<Stanford Research Systems|SR850 DSP|cec817f5>  
ConnectionRecord<HP|3478A|bd92c887>
```

or filter by the equipment that use GPIB as the communication bus and that are from Hewlett Packard<sup>1</sup>

```
>>> for conn in db.connections(address='GPIB', manufacturer='H.*P'):  
...     print(conn)  
...  
ConnectionRecord<HP|3478A|bd92c887>
```

Access the *EquipmentRecord* that has the specified alias *dmm* in the configuration file

```
>>> record = db.equipment['dmm']  
>>> print(record)  
EquipmentRecord<Hewlett Packard|34401A|15ab8c6c>  
>>> record.is_calibration_due()  
False  
>>> print(record.connection)  
ConnectionRecord<Hewlett Packard|34401A|15ab8c6c>
```

Establishing a connection to the equipment is achieved by calling the *connect()* method of an *EquipmentRecord*. This will return a specific *Connection* subclass that contains the necessary properties and methods for communicating with the equipment.

```
>>> dmm = record.connect()  
>>> dmm.query('*IDN?')  
'Hewlett Packard,34401A,15ab8c6c,A.02.14-02.40-02.14-00.49-03-01\n'
```

## CONTENTS

### 1.1 Configuration File

A configuration file is used by MSL-Equipment to:

1. Specify which *Databases* to use
2. Specify the equipment that is being used to perform a measurement
3. Specify additional parameters to use in your program

The configuration file uses the eXtensible Markup Language (XML) format to specify this information.

The following illustrates an example configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<msl>
  <!-- OPTIONAL: Use PyVISA-py as the PyVISA backend library.
  Other values are:
    @ivi (PyVISA >=1.11)
    @ni (PyVISA <1.11)
    @sim (PyVISA-sim) -->
  <pyvisa_library>@py</pyvisa_library>

  <!-- OPTIONAL: Whether to open all connections in demo mode. -->
  <demo_mode>true</demo_mode>

  <!-- OPTIONAL: Add paths to where external resource files are located. The
  paths get appended to Config.PATH and os.environ['PATH']. If the recursive=
  ↪ "true"
  ↪ attribute is included then recursively adds all sub-directories starting
  ↪ from
  ↪ the root directory (also includes the root directory). -->
  <path>I:\Photometry\SDKs</path>
  <path recursive="true">D:\code\resources\lib</path>

  <!-- OPTIONAL: The user can define their own elements. -->
  <max_temperature units="C">60</max_temperature>

  <!-- OPTIONAL: Specify the equipment that is being used to perform the
  ↪ measurement
  ↪ and assign an "alias" that you want to use to associate for each
```

(continues on next page)

(continued from previous page)

```

↪equipment. You
    only need to specify enough XML attributes to uniquely identify the
↪equipment record
    in an Equipment-Registry database. For example, if there is only 1
↪equipment record
    in the Equipment-Registry databases (see the <registers> tag below) that
↪is from
    "Company XYZ" then specifying manufacturer="Company XYZ" is enough
↪information to
    uniquely identify the equipment record. If in the future another
↪equipment record
    is added to an Equipment-Registry database for "Company XYZ" then an
↪exception will
    be raised telling you to specify more information in the configuration
↪file to
    uniquely identify a single equipment record. -->
<equipment alias="dmm" manufacturer="Keysight" model="34465A"/>
<equipment alias="scope" manufacturer="Pico Technologies"/>
<equipment alias="flipper" manufacturer="Thorlabs" model="MFF101/M" serial=
↪"123456"/>

<!-- REQUIRED: Specify the Equipment-Register Databases to load equipment
↪records from. -->
<registers>
  <!-- The "team" attribute is used to specify which research team the
    Equipment-Register database belongs to. -->
  <register team="P&R">
    <path>Z:\Equipment\Equipment Register.xls</path>
    <!-- If there are multiple Sheets in the Excel database then you must
↪specify the
    name of the Sheet that contains the equipment records. This Excel
↪database
    also contains connection records (see the <connections> tag below)
↪and so
    the <sheet> tag must be specified. -->
    <sheet>Equipment</sheet>
  </register>
  <register team="Electrical">
    <path>H:\Quality\Registers\Equipment.xlsx</path>
    <!-- No need to specify the Sheet name if there is only 1 Sheet in the
↪Excel database. -->
  </register>
  <register team="Pressure">
    <!-- Using the XML or JSON format to store equipment records allows
↪for including
    MaintenanceRecords, CalibrationRecords, MeasurandRecords, ... -->
    <path>J:\Registers\Equipment.xml</path>
  </register>
  <!-- For a text-based database (e.g., CSV, TXT files) you can specify

```

(continues on next page)



(continued from previous page)

```

↪how the dates are
    formatted and the encoding that is used in the file (UTF-8 is assumed.
↪if the encoding
    is not specified). A CSV database uses "," as the delimiter and a TXT
↪database uses
    "\t" as the delimiter. -->
    <register team="Time" date_format="%d.%m.%y" encoding="cp1252">
      <path>W:\Registers\Equip.csv</path>
    </register>
    <register team="Mass" date_format="%Y-%m-%d">
      <!-- You can also specify the database path to be a path that is
↪relative to the
        location of the configuration file. For example, this "equip-reg.txt"
↪file is
        located in the same directory as the configuration file. -->
      <path>equip-reg.txt</path>
    </register>
    <register team="Length" user_defined="apples, pears, oranges">
      <!-- An EquipmentRecord has standard properties (e.g, manufacturer,
↪model, ...) that
        are read from the database. You can also include additional fields
↪from the database
        that are not part of the standard properties. Include a "user_defined
↪" list
        (comma-separated) of additional properties to include. The field
↪names that
        contain the text "apples", "pears" and "oranges" are added to the
↪"user_defined"
        dictionary for all EquipmentRecord's in this register. -->
      <path>I:\LS-Equip-Reg\reg.csv</path>
    </register>
  </registers>

  <!-- OPTIONAL: Specify the Connection Databases to load connection records
↪from. -->
  <connections>
    <connection>
      <path>Z:\Equipment\Equipment Register.xls</path>
      <!-- Must also specify which Sheet in this Excel database contains the
↪connection records.
        This "Equipment Register.xls" file also contains an "Equipment" Sheet,
↪ see the
        <register team="P&R"> element above. -->
      <sheet>Connections</sheet>
    </connection>
    <!-- You can set the encoding that is used for a text-based database. -->
    <connection encoding="utf-16">
      <!-- Specify a relative path (relative to the location of the
↪configuration file). -->

```

(continues on next page)

(continued from previous page)

```
<path>data/my_connections.txt</path>
</connection>
</connections>

</msl>
```

The *Config* class is used to load a configuration file and it is the main entry point, for example

```
>>> from msl.equipment import Config
>>> cfg = Config('config.xml')
```

## 1.2 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 **xml**, **json**, **txt** (\t delimited), **csv** (, delimited) and **xls[x]**.

The **txt**, **csv** and **xls[x]** formats are simple databases that are composed of *fields* (columns) and *records* (rows). The **xml** and **json** formats allow for storing more complex data structures, such as *MaintenanceRecord*'s *CalibrationRecord*'s and *MeasurandRecord*'s for each *EquipmentRecord*. For example **xml** and **json** formats, see *equipment\_register.xml* and *equipment\_register.json* respectively.

### 1.2.1 Equipment-Register Database

**Attention:** The design of the Equipment-Register database is in active development and it will be unstable until an official release of MSL-Equipment is made.

The information about the equipment that is used to perform a measurement must be known and it must be kept up to date. Keeping a central and official (hence the word *Register*) database of the equipment that is available in the laboratory allows for easily managing this information and for helping to ensure that the equipment that is being used for a measurement meets the calibration requirements needed to obtain the desired measurement uncertainty.

MSL-Equipment does not require that a *single* database is used for all equipment records. However, it is vital that each equipment record can only be uniquely found in one *Equipment-Register Database*. The records in a database must never be copied from one database to another database (*keeping a backup copy of the database is encouraged*). Rather, if you are borrowing equipment from another team you simply specify the path to that team's *Equipment-Register Database* as a **<register>** element in your *Configuration File*. The owner of the equipment is responsible for ensuring that the information about the equipment is kept up to date in their *Equipment-Register Database* and the user of the equipment defines an **<equipment>** element in the *Configuration File* to access this information. Therefore, an *Equipment-Register Database* is to be considered as a *global* database that can be accessed (with read permission only) by anyone.

Each record in an *Equipment-Register Database* is converted into an *EquipmentRecord*.

The following is an example of an *Equipment-Register Database* (additional *fields* can also be added to a database, see *Field Names*).

| Manufacturer    | Model Number | Serial Number | Description                           |
|-----------------|--------------|---------------|---------------------------------------|
| Keysight        | 34465A       | MY5450        | 6.5 digit digital multimeter          |
| Hewlett Packard | HP8478B      | BCD024        | Dual element thermistor power sensors |
| Agilent         | 53230A       | 49e39f        | Universal counter/timer               |

**Tip:** Not all records in the *Equipment-Register Database* need to have the ability to be interfaced with a computer. For example, cables, amplifiers, filters and adaptors can all be important equipment that may be used to perform a measurement and should be included in the *Equipment-Register Database* and specified as <equipment> elements in the *Configuration File*.

## Field Names

Some of the supported *fields* for an *Equipment-Register Database* are:

- **Category** – The category (e.g., Laser, DMM) that the equipment belongs to.
- **Description** – A description of the equipment.
- **Location** – The location where the equipment can usually be found.
- **Manufacturer** – The name of the manufacturer of the equipment.
- **Model** – The model number of the equipment.
- **Serial** – The serial number, or engraved unique ID, of the equipment.

The text in the header of each *field* is not too particular for what it must be. The header text is parsed for one of the specific *field* names listed above and if the header contains one of these *field* names then that column is assigned to be that *field*.

For example, the following headers are valid (the blue text is what is important in the header)

- Headers can contain many words. For a *field* to be assigned to the *manufacturer* attribute the header can be written as:

*This column is used to specify the Manufacturer of the equipment*

- Text is case insensitive. For a *field* to be assigned to the *model* attribute the header can be written as any of the following:

MODEL No.    Model #    The model number of the equipment    MoDeL

Although using the following header will not raise an exception, you should not use the following header because either the *manufacturer* or the *model* attribute will be assigned for this *field* depending on the order in which the *fields* appear in the database:

*The model number given by the manufacturer*

- Whitespace is replaced by an underscore. For a *field* to be assigned to the *is\_operable* attribute the header can be written as:

|                                   |
|-----------------------------------|
| Is Operable, <i>True or False</i> |
|-----------------------------------|

- If the header does not contain any of the specific *field* names that are being searched for then the values in that column are silently ignored.

Equipment records should be defined in an *Equipment-Register Database* and accessed via the *database()* method; however, you can also define equipment records in a Python module, for example:

```
from msl.equipment import EquipmentRecord, ConnectionRecord, Backend

equipment = {
    'dmm':
        EquipmentRecord(
            manufacturer='HP',
            model='34401A',
            serial='123456789',
            connection=ConnectionRecord(
                backend=Backend.MSL,
                address='COM3',
                properties=dict(
                    baud_rate=19200,
                )
            )
        ),
    'scope':
        EquipmentRecord(
            manufacturer='Pico Technology',
            model='5244B',
            serial='XY135/001',
            description='Oscilloscope -- 2 Channel, 200 MHz, 1 GSPS, 512 Mpts,
↪ 5.8 ns',
            connection=ConnectionRecord(
                backend=Backend.MSL,
                address='SDK:ps5000a.dll',
                properties=dict(
                    resolution='16bit',
                )
            )
        ),
    '1ohm':
        EquipmentRecord(
            manufacturer='Tinsley',
            model='64750',
            serial='5672413',
            description='1.0 Ohm Resistor 3A',
        ),
}
```

## 1.2.2 Connections Database

A *Connections Database* is used to store the information that is required to establish communication with the equipment.

You specify the *Connections Database* that you want to use as a <connection> element in your *Configuration File*. Each record in an *Connections Database* is converted into a *ConnectionRecord*.

### Field Names

The supported *fields* for a *Connections Database* are:

- **Address** – The address to use for the connection (see *Address Syntax*).
- **Backend** – The *Backend* to use to communicate with the equipment.
- **Manufacturer** – The name of the manufacturer of the equipment.
- **Model** – The model number of the equipment.
- **Properties** – Additional properties that may be required to establish a connection to the equipment as key-value pairs separated by a semi-colon. For example, for a *ConnectionSerial* connection the baud rate and parity might need to be defined – baud\_rate=11920; parity=even. The value (as in a key-value pair) gets cast to the appropriate data type (e.g., *int*, *float*, *str*) so the baud rate value would be 11920 as an *int* and the parity value becomes *Parity.EVEN*.
- **Serial** – The serial number, or unique ID, of the equipment.

A record in a *Connections Database* gets matched with the appropriate record in an *Equipment-Register Database* by the unique combination of the manufacturer + model + serial values, which when combined act as the primary key in each database.

The following is an example of a *Connections Database*. The header of each *field* also follows the same *Field Names* format used in an *Equipment-Register Database* and so *MODEL#* would also be an acceptable header for *Model Number*.

| Manufacturer    | Model Number | Serial Number | Backend | Address             | Properties                    |
|-----------------|--------------|---------------|---------|---------------------|-------------------------------|
| OMEGA           | iTHX-W3      | 458615        | MSL     | TCP::192.168.1.100: | termination="r"; time-out=10  |
| Hewlett Packard | 3468A        | BCD024        | PyVISA  | GPIOB::7            |                               |
| Agilent         | 53230A       | 49e39f        | MSL     | COM2                | baud_rate=119200; parity=even |

Unlike an *Equipment-Register Database* each person can maintain their own *Connections Database*. The reason being that since equipment can be shared between people some Connection *fields*, like the COM address, can vary depending on which computer the equipment is connected to and what other equipment is also connected to that computer. Therefore, everyone could have their own *Connections Database* and connection records can be copied from one *Connections Database* to another. If you are using someone else's equipment and if none of the Connection *fields* need to be changed to be able to communicate with the equipment then it is recommended to add their *Connections Database* as a <connection> element in your *Configuration File*.

## Address Syntax

The following are examples of an **Address** syntax (see more examples from [National Instruments](#)).

| Interface     | Syntax Example                              | Notes  |
|---------------|---|--|
| PRO-LOGIX     | Pro-logix::192.168.1.110::1234              | The GPIB-ETHERNET Controller: host=192.168.1.110, port=1234, primary-GPIB-address=6  |
| PRO-LOGIX     | Pro-logix::192.168.1.70::1234               | The GPIB-ETHERNET Controller: host=192.168.1.70, port=1234, primary-GPIB-address=6, secondary-GPIB-address=112   |
| PRO-LOGIX     | Pro-logix::192.168.1.70::1234               | The GPIB-ETHERNET Controller: host=192.168.1.70, port=1234, primary-GPIB-address=6, secondary-GPIB-address=112   |
| PRO-LOGIX     | Prologix::COM3::6                           | The GPIB-USB Controller: port=COM3, primary-GPIB-address=6   |
| PRO-LOGIX     | Pro-logix::/dev/ttyS0::4::96                | The GPIB-USB Controller: port=/dev/ttyS0, primary-GPIB-address=4, secondary-GPIB-address=96  |
| SDK           | SDK::C:/Program Files/Manufacturer/bin/file | Specify the full path to the SDK   |
| SDK           | SDK::filename.dll                           | Specify only the filename if the path to where the SDK file is located has been added as a <path> element in the <i>Configuration File</i>               |
| SE-RIAL       | COM2  | A serial port on Windows   |
| SE-RIAL       | ASRL/dev/ttyS1                              | A serial port on Linux   |
| SE-RIAL       | ASRL2::INSTR                                | Compatible with <a href="#">National Instruments</a> syntax  |
| SE-RIAL       | ASRLCOM2                                    | Compatible with <a href="#">PyVISA-py</a> syntax   |
| SOCKE'        | TCP::192.168.1.100::5000                    | Creates the connection as a <code>socket.SOCK_STREAM</code> to the IP address <b>192.168.1.100</b> at port <b>5000</b>                                   |
| SOCKE'        | UDP::192.168.1.100::5000                    | Creates the connection as a <code>socket.SOCK_DGRAM</code>   |
| SOCKE'        | TCPIP::192.168.1.100::50                    | Compatible with <a href="#">National Instruments</a> syntax  |
| SOCKE'        | SOCKET::192.168.1.100::                     | Generic socket type. You can specify the connection type in the <i>Properties field</i> (i.e., type=RAW)   |
| TCPIP HiS-LIP | TCPIP::dev.company.com:                     | A HiSLIP LAN instrument at the hostname <b>dev.company.com</b> .   |
| TCPIP HiS-LIP | TCPIP::10.12.114.50::hisl                   | A HiSLIP LAN instrument whose IP address is <b>10.12.114.50</b> with the server listening at port <b>5000</b>  |
| TCPIP VXI-11  | TCPIP::dev.company.com:                     | A VXI-11.3 LAN instrument at the hostname <b>dev.company.com</b> . This uses the default LAN Device Name <b>inst0</b>                                    |
| TCPIP VXI-11  | TCPIP::10.6.56.21::gpi0,                    | A VXI-11.2 GPIB device whose IP address is <b>10.6.56.21</b>   |
| TCPIP VXI-11  | TCPIP::192.168.1.100                        | A VXI-11.3 LAN instrument at IP address <b>192.168.1.100</b> . Note that default values for board <b>0</b> and LAN device name <b>inst0</b> will be used |
| ZMQ           | ZMQ::192.168.20.90::555                     | Use the <a href="#">ZeroMQ</a> messaging library to connect to a device at IP address <b>192.168.1.100</b> and port <b>5555</b>                          |

## 1.3 Install MSL-Equipment

To install **MSL-Equipment** run

```
pip install https://github.com/MSLNZ/msl-equipment/archive/v0.1.0.zip#egg=msl-  
↪equipment
```

Alternatively, using the [MSL Package Manager](#) run

```
msl install equipment
```

### 1.3.1 Dependencies

- Python 2.7, 3.5+
- [msl-loadlib](#)
- [msl-io](#)
- [numpy](#)
- [pyserial](#)
- [python-dateutil](#)
- [pyzmq](#)

### 1.3.2 Optional Dependencies

- [PyVISA](#)
- [PyVISA-py](#)
- [NI-DAQmx](#)

Some of the [MSL Resources](#) might not work in your application because the resource might depend on an external dependency (e.g., the SDK provided by a manufacturer) and this external dependency might not be available for your operating system.

## 1.4 Examples

Some example scripts that illustrate how to use MSL-Equipment to communicate with equipment can be found in the [repository](#).



## 1.5 MSL Resources

MSL Resources are specific classes that are used to communicate with the equipment.

- [Aim and Thurlby Thandar Instruments](#)
  - [MXSeries](#) – MX100QP, MX100TP, MX180TP
- [Avantes](#)
  - [Avantes](#) – Wrapper around the [AvaSpec SDK](#)
- [Bentham Instruments Ltd](#)
  - [Bentham](#) – Wrapper around the [Bentham SDK](#)
- [CMI \(Czech Metrology Institute\)](#)
  - [SIA3](#) – Switched Integrator Amplifier
- [DataRay](#)
  - [DataRayOCX64](#) – Wrapper around the [DATARAYOCX](#) library
- [Electron Dynamics Ltd](#)
  - [TCSeries](#) – Temperature Controller (TC LV, TC M, TC Lite)
- [Energetiq](#)
  - [EQ99](#) – EQ-99 Manager
- [MKS Instruments](#)
  - [PR4000B](#) – Flow and Pressure controller
- [NKT Photonics](#)
  - [NKT](#) – Wrapper around the [NKT Photonics SDK](#)
- [OMEGA](#)
  - [iTHX](#) – iTHX-W3, iTHX-D3, iTHX-SD, iTHX-M, iTHX-W, iTHX-2
- [OptoSigma](#)
  - [SHOT702](#) – Two-axis Stage Controller
- [Optronic Laboratories](#)
  - [OL756](#) – UV-VIS spectroradiometer
  - [OLCurrentSource](#) – DC Current Source (16A, 65A, 83A)
- [Pico Technology](#) – Wrapper around the [Pico Technology SDK](#)
  - [PicoScope](#)
    - \* [PicoScope2000](#) - PicoScope 2000 Series
    - \* [PicoScope2000A](#) - PicoScope 2000 Series A
    - \* [PicoScope3000](#) - PicoScope 3000 Series
    - \* [PicoScope3000A](#) - PicoScope 3000 Series A
    - \* [PicoScope4000](#) - PicoScope 4000 Series

- \* *PicoScope4000A* - PicoScope 4000 Series A
- \* *PicoScope5000* - PicoScope 5000 Series
- \* *PicoScope5000A* - PicoScope 5000 Series A
- \* *PicoScope6000* - PicoScope 6000 Series
- PT-104 Platinum Resistance Data Logger
  - \* *PT104* – PT-104
- Princeton Instruments
  - *PrincetonInstruments* – Wrapper around the *ARC\_Instrument.dll* library
- Raicol Crystals
  - *RaicolTEC* – TEC (Peltier-based) oven
- Thorlabs
  - Wrapper around the *Kinesis* SDK.
    - \* *FilterFlipper* – MFF101, MFF102
    - \* *IntegratedStepperMotors* – LTS150, LTS300, MLJ050, MLJ150, K10CR1
    - \* *KCubeSolenoid* – KSC101
    - \* *KCubeStepperMotor* – KST101
    - \* *KCubeDCServo* – KDC101
    - \* *BenchtopStepperMotor* – BSC101, BSC102, BSC103, BSC201, BSC202, BSC203
  - *FilterWheelXX2C* – FW102C, FW212C

### 1.5.1 Creating a new MSL Resource

When adding a new MSL Resource class to the *repository* the following steps should be performed. Please follow the *style guide*.

---

**Note:** If you do not want to upload the new MSL Resource class to the *repository* then you only need to write the code found in Step 5 to use your class in your own programs.

---

1. Create a *fork* of the *repository*.
2. If you are adding a new MSL Resource for equipment from a manufacturer that does not already exist in the *msl/equipment/resources* directory then create a new Python package in *msl/equipment/resources* using the name of the *manufacturer* as the package name.
3. Create a new Python module, in the package from Step 2, using the *model number* of the equipment as the name of the module.
4. If a *msl.equipment.exceptions* class has not been created for this manufacture then create a new exception handler class using the name of the *manufacturer* in the class name.
5. Create a new connection class within the module that you created in Step 3. The class must be a subclass of one of the *Connection Classes*.

```
# msl/equipment/resources/<manufacturer>/<model_number>.py
#
from msl.equipment.resources import register
from msl.equipment.exceptions import TheErrorClassFromStep4 # this is
↳ optional
from msl.equipment.connection_xxx import ConnectionXxx # replace xxx
↳ with the Connection subclass

# Register your class so that MSL-Equipment knows that it exists
@register(manufacturer='a regex pattern', model='a regex pattern') # can
↳ include a `flags` kwarg
class ModelNumber(ConnectionXxx): # change ModelNumber and ConnectionXxx

    def __init__(self, record):
        """Edit the docstring...

        Do not instantiate this class directly. Use the :meth:`~.
↳ EquipmentRecord.connect`
        method to connect to the equipment.

        Parameters
        -----
        record : :class:`~.EquipmentRecord`
            A record from an :ref:`equipment-database`.
        """
        super(ModelNumber, self).__init__(record) # change ModelNumber

        # the following is optional, the default exception handler is
↳ MSLConnectionError
        self.set_exception_class(TheErrorClassFromStep4) # change
↳ TheErrorClassFromStep4
```

6. Add at least one example for how to use the new MSL Resource in `msl/examples/equipment`. Follow the template of the other examples in this package for naming conventions and for showing how to use the new MSL Resource.
7. Create tests for the new MSL Resource. The tests cannot be dependent on whether the equipment is physically connected to the computer running the test (ideally the examples that you write in Step 6 will demonstrate that communicating with the equipment works). The very minimal test to create is to add a test case to the `def test_find_resource_class()` function for ensuring that your class is returned for various values of *manufacturer* and *model*. Run the tests using `python setup.py test` (ideally you would run the tests for all *currently-supported versions* of Python, see also `conda-tests.py`).
8. Add `.rst` documentation files for the new MSL Resource to the `docs/_api` folder. You can either run `python setup.py apidoc` to automatically generate the `.rst` documentation files or you can create the necessary `.rst` files manually. Running `python setup.py apidoc` will generate `.rst` files for *all* modules in **MSL-Equipment** in the `docs/_autosummary` folder. Only copy the `.rst` files that are associated with your new MSL Resource to the `docs/_api` folder. After copying the files you can delete the `docs/_autosummary` folder before running `python setup.py docs` to build the documentation, otherwise you will get numerous warnings. If you want to manually create the `.rst` files then look in the `docs/_api` folder for examples from other MSL Resources.

9. If you created a new package in Step 2 then you need to add the new package to the `toctree` of the Subpackages section in `docs/_api/msl.equipment.resources.rst`. Insert the name of the new MSL Resource package in the file alphabetically. If you forget to do this step then a warning will appear when building the documentation to help remind you to do it. If you did not create a new package in Step 2 then add the `.rst` file from Step 8 to the Subpackages section in the appropriate `msl.equipment.resources.*.rst` file.
10. Add the new MSL Resource class, alphabetically, to the list of MSL Resources in `docs/resources.rst`. Follow the template that is used for the other MSL Resources listed in this file.
11. Add yourself to `AUTHORS.rst` and add a note in `CHANGES.rst` that you created this new Resource. These files are located in the root directory of the **MSL-Equipment** package.
12. If running the tests pass and building the docs show no errors/warnings then create a [pull request](#).

## 1.6 API Documentation

Although this package contains many classes and functions, the only object that you must initialize in your application is `Config` and perhaps `EquipmentRecord`'s (depending on the format that is chosen to store the `Databases`).

### 1.6.1 Connection Classes

Use the following functions to find equipment that are connected to a computer or that are on the network

|                                |   |
|--------------------------------|---|
| <code>list_resources()</code>  | A <code>dict</code> of all equipment that are available to connect to |
| <code>print_resources()</code> | Print a summary of all equipment that are available to connect to     |

The following `Connection` classes are available to communicate with the equipment (*although you should never need to instantiate these classes directly*):

|                                     |  |
|-------------------------------------|--|
| <code>ConnectionDemo</code>         | Simulate a connection to the equipment                                 |
| <code>ConnectionMessageBased</code> | Equipment that use message-based communication                         |
| <code>ConnectionPrologix</code>     | Equipment that is connected through a <code>Prologix</code> Controller |
| <code>ConnectionSDK</code>          | Equipment that use the manufacturer's SDK for the connection           |
| <code>ConnectionSerial</code>       | Equipment that is connected through a Serial port                      |
| <code>ConnectionSocket</code>       | Equipment that is connected through a Socket                           |
| <code>ConnectionTCPIP VXI11</code>  | Equipment that use the <code>VXI-11</code> protocol                    |
| <code>ConnectionTCIP HiSLIP</code>  | Equipment that use the <code>HiSLIP</code> protocol                    |
| <code>ConnectionZeroMQ</code>       | Equipment that use the <code>ZeroMQ</code> protocol                    |

and the `Connection` classes that are available from external Python libraries are:

|                               |   |
|-------------------------------|---|
| <code>ConnectionPyVISA</code> | Uses <code>PyVISA</code> to establish a connection to the equipment |
| <code>ConnectionNIDAQ</code>  | Uses <code>NI-DAQ</code> to establish a connection to the equipment |

## 1.6.2 Package Structure

### **msl.equipment package**

Manage and connect to equipment in the laboratory.

`msl.equipment.version_info = version_info(major=0, minor=1, micro=0, releaselevel='final')`

Contains the version information as a (major, minor, micro, releaselevel) tuple.

#### **Type**

`namedtuple`

`msl.equipment.list_resources(hosts=None, timeout=2)`

Returns a dictionary of all equipment that are available to connect to.

#### **Parameters**

- **hosts** (`list` of `str`, optional) – The IP address(es) on the computer to use to find network devices. If not specified, then find devices on all network interfaces.
- **timeout** (`float`, optional) – The maximum number of seconds to wait for a reply from a network device.

#### **Returns**

`dict` – The information about the devices that were found.

`msl.equipment.print_resources(**kwargs)`

Print a summary of all equipment that are available to connect to.

#### **Parameters**

**kwargs** – All keyword arguments are passed to `list_resources()`.

### **msl.equipment.config module**

Load an XML *Configuration File*.

**class** `msl.equipment.config.Config(path)`

Bases: `object`

Load an XML *Configuration File*.

This function is used to set the configuration constants to use for the Python runtime and it allows you to access *EquipmentRecord*'s from an *Equipment-Register Database* and *ConnectionRecord*'s from a *Connections Database*.

The following table summarizes the XML elements that are used by **MSL-Equipment** and can be defined in a *Configuration File*:

| XML Tag | Examples                               | Values | Description  |
|---------|--|--------|--|
| pyvisa_ | @ivi,<br>@sim,<br>/path/to/libvisa.so. | @py,   | The PyVISA <a href="#">library</a> to use.   |
| demo_n  | true, false, True,<br>False            |        | Whether to open connections in demo mode.  |
| path    | /path/to/SDKs,<br>D:/images            |        | A path that contains external resources. Accepts a <i>recursive="true"</i> attribute. The path(s) are appended to <code>os.environ['PATH']</code> and to <i>PATH</i> |

The user is also encouraged to define their own application-specific elements within the *Configuration File*.

**Parameters**

**path** ([str](#)) – The path to an XML *Configuration File*.

**Raises**

**OSError** – If *path* does not exist or if the *Configuration File* is invalid.

**PyVISA\_LIBRARY = '@ivi'**

The PyVISA backend [library](#) to use.

**Type**

[str](#)

**DEMO\_MODE = False**

Whether to open connections in demo mode.

If enabled then the equipment does not need to be physically connected to a computer and the connection is simulated.

**Type**

[bool](#)

**PATH = []**

Paths are also appended to `os.environ['PATH']`.

**Type**

[list of str](#)

**property path**

The path to the configuration file.

**Type**

[str](#)

**property 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*, *default=None*)

Gets the value associated with the specified *tag* in the configuration file.

The first element with name *tag* (relative to the *root*) is used.

The value is converted to the appropriate data type if possible. Otherwise, the value will be returned as a *str*.

**Parameters**

- **tag** (*str*) – The name of an XML tag in the configuration file.
- **default** – The default value if *tag* cannot be found.

**Returns**

The value associated with *tag* or *default* if the tag cannot be found.

**find**(*tag*)

Find the first sub-element (from the *root*) matching *tag* in the configuration file.

**Parameters**

**tag** (*str*) – The name of an XML tag in the configuration file.

**Returns**

*Element* or *None* – The first sub-element or *None* if the tag cannot be found.

**findall**(*tag*)

Find all matching sub-elements (from the *root*) matching *tag* in the configuration file.

**Parameters**

**tag** (*str*) – The name of an XML tag in the configuration file.

**Returns**

*list* of *Element* – All matching elements in document order.

**attrib**(*tag*)

Get the attributes of an *Element* in the configuration file.

The first element with name *tag* (relative to the *root*) is used.

The values are converted to the appropriate data type if possible. Otherwise, the value will be kept as a *str*.

**Parameters**

**tag** (*str*) – The name of an XML element in the configuration file.

**Returns**

*dict* – The attributes of the *Element*. If an element with the name *tag* does not exist, then an empty *dict* is returned.

## msl.equipment.connection module

Base class for establishing a connection to the equipment.

**class** msl.equipment.connection.**Connection**(*record*)

Bases: `object`

Base class for establishing a connection to the equipment.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**property** `equipment_record`

The information about the equipment.

### Type

*EquipmentRecord*

**disconnect**()

Disconnect from the equipment.

This method should be overridden in the 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 garbage collected, which happens when the reference count is 0.

---

**raise\_exception**(*message*)

Raise an `MSLConnectionError` and log the error message.

### Parameters

**message** (`str` or `Exception`) – The message to display in the exception class that was set by `set_exception_class()`. If an `Exception` object then its string representation is used as the message.

**static** `convert_to_enum`(*obj*, *enum*, *prefix=None*, *to\_upper=False*, *strict=True*)

Convert *obj* to an Enum.

### Parameters

- **obj** – Any object to be converted to the specified *enum*. Can be a value of member of the specified *enum*.
- **enum** – The `Enum` object that *obj* should be converted to.
- **prefix** (`str`, optional) – If *obj* is a `str`, then ensures that *prefix* is included at the beginning of *obj* before converting *obj* to the *enum*.



- **to\_upper** (*bool*, optional) – If *obj* is a *str*, then whether to change *obj* to be upper case before converting *obj* to the *enum*.
- **strict** (*bool*, optional) – Whether errors should be raised. If *False* and *obj* cannot be converted to *enum* then *obj* is returned and the error is logged.

**Returns**

*Enum* – The *enum*.

**Raises**

**ValueError** – If *obj* is not in *enum* and *strict* is *True*.

**static log\_debug**(*msg*, \**args*, \*\**kwargs*)

Log a debug message.

All input parameters are passed to *debug()*.

**static log\_info**(*msg*, \**args*, \*\**kwargs*)

Log an info message.

All input parameters are passed to *info()*.

**static log\_warning**(*msg*, \**args*, \*\**kwargs*)

Log a warning message.

All input parameters are passed to *warning()*.

**static log\_error**(*msg*, \**args*, \*\**kwargs*)

Log an error message.

All input parameters are passed to *error()*.

**static log\_critical**(*msg*, \**args*, \*\**kwargs*)

Log a critical message.

All input parameters are passed to *critical()*.

**set\_exception\_class**(*handler*)

Set the exception-handler class for this *Connection*.

**Parameters**

**handler** (*MSLConnectionError*) – A subclass of *MSLConnectionError*

**Raises**

**TypeError** – If the *handler* is not a subclass of *MSLConnectionError*

**static parse\_address**(*address*)

Determine whether a subclass should be used to connect to the equipment.

**Attention:** The subclass should override this method.

**Parameters**

**address** (*str*) – The address of a *ConnectionRecord*.

**Returns**

*dict* or *None* – If the *address* is in a valid format for the subclass to be able to connect to the equipment then a *dict* is returned containing the information

necessary to connect to the equipment. Otherwise, if the *address* is not valid for the subclass to be able to connect to the equipment then `None` is returned.

### msl.equipment.connection\_demo module

Simulate a connection to the equipment.

**class** msl.equipment.connection\_demo.**ConnectionDemo**(*record*, *cls*)

Bases: `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 a 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 method. For example, if `:rtype: int` then an `int` is returned or if `:rtype: int, float` then an `int` and a `float` are 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 `record.connect(demo=True)` method to connect to the equipment in demo mode or set `DEMO_MODE` to be `True` in the *Configuration File* to open all connections in demo mode.

#### Parameters

- **record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.
- **cls** (`Connection`) – A `Connection` subclass (that has **NOT** been instantiated).

**disconnect()**

Log a disconnection from the equipment.

### msl.equipment.connection\_message\_based module

Base class for equipment that use message-based communication.

**class** msl.equipment.connection\_message\_based.**ConnectionMessageBased**(*record*)

Bases: `Connection`

Base class for equipment that use message-based communication.

The *backend* value must be equal to `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 `connect()` method to connect to the equipment.

#### Parameters

- **record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**CR = b'\r'**

The carriage-return character (hex: 0x0D, decimal: 13).

**Type**

bytes

**LF = b'\n'**

The line-feed character (hex: 0x0A, decimal: 10).

**Type**

bytes

**property encoding**

The encoding that is used for `read()` and `write()` operations.

**Type**

str

**property encoding\_errors**

The error handling scheme to use when encoding and decoding messages.

For example: *strict*, *ignore*, *replace*, *xmlcharrefreplace*, *backslashreplace*

**Type**

str

**property read\_termination**

The termination character sequence that is used for the `read()` method.

Reading stops when the equipment stops sending data or the `read_termination` character sequence is detected. If you set the `read_termination` to be equal to a variable of type `str` it will automatically be encoded.

**Type**

bytes or None

**property write\_termination**

The termination character sequence that is appended to `write()` messages.

If you set the `write_termination` to be equal to a variable of type `str` it will automatically be encoded.

**Type**

bytes or None

**property max\_read\_size**

The maximum number of bytes that can be `read()`.

**Type**

int

**property timeout**

The timeout, in seconds, for `read()` and `write()` operations.

A value 0 will set the timeout to be `None` (blocking mode).

**Type**

float or None

**property rstrip**

Whether to remove trailing whitespace from `read()` messages.

**Type**

`bool`

**raise\_timeout**(*append\_msg=""*)

Raise a `MSLTimeoutError`.

**Parameters**

**append\_msg** (`str`, optional) – A message to append to the generic timeout message.

**read**(*size=None, fmt='ascii', dtype=None, decode=True*)

Read a message from the equipment.

This method will block until one of the following conditions is fulfilled:

1. the `read_termination` byte(s) is(are) received – only if `read_termination` is not `None`.
2. `size` bytes have been received – only if `size` is not `None`.
3. a timeout occurs – only if `timeout` is not `None`. An `MSLTimeoutError` is raised.
4. `max_read_size` bytes have been received. An `MSLConnectionError` is raised.

**Parameters**

- **size** (`int`, optional) – The number of bytes to read. Ignored if it is `None`.
- **fmt** (`str` or `None`, optional) – The format that the message data is in. Ignored if `dtype` is not specified. See `from_bytes()` for more details.
- **dtype** – The data type of the elements in the message data. Can be any object that `numpy.dtype` supports. See `from_bytes()` for more details. For messages that are of scalar type (i.e., a single number) it is more efficient to not specify `dtype` but to pass the message to the `int` or `float` class to convert the message to the appropriate numeric type.
- **decode** (`bool`, optional) – Whether to decode the message (i.e., convert the message to a `str`) or keep the message as `bytes`. Ignored if `dtype` is specified.

**Returns**

`str`, `bytes` or `ndarray` – The message from the equipment. If `dtype` is specified, then the message is returned as an `ndarray`, if `decode` is `True` then the message is returned as a `str`, otherwise the message is returned as `bytes`.

**See also:**

`rstrip`

**write**(*message, data=None, fmt='ieee', dtype='<f'*)

Write a message to the equipment.

**Parameters**

- **message** (`str` or `bytes`) – The message to write to the equipment.

- **data** (`list`, `tuple` or `numpy.ndarray`, optional) – The data to append to *message*. See `to_bytes()` for more details.
- **fmt** (`str` or `None`, optional) – The format to use to convert *data* to bytes. Ignored if *data* is `None`. See `to_bytes()` for more details.
- **dtype** – The data type to use to convert each element in *data* to bytes. Ignored if *data* is `None`. See `to_bytes()` for more details.

**Returns**

`int` – The number of bytes written.

**query**(*message*, *delay*=0.0, *\*\*kwargs*)

Convenience method for performing a `write()` followed by a `read()`.

**Parameters**

- **message** (`str` or `bytes`) – The message to write to the equipment.
- **delay** (`float`, optional) – The time delay, in seconds, to wait between `write()` and `read()` operations.
- **\*\*kwargs** – All additional keyword arguments are passed to `read()`

**Returns**

`str`, `bytes` or `ndarray` – The message from the equipment. If *dtype* is specified, then the message is returned as an `ndarray`, if *decode* is `True` then the message is returned as a `str`, otherwise the message is returned as `bytes`.

**msl.equipment.connection\_nidaq module**

Uses `NI-DAQ` as the backend to communicate with the equipment.

**class** `msl.equipment.connection_nidaq.ConnectionNIDAQ`(*record*)

Bases: `Connection`

Uses `NI-DAQ` to establish a connection to the equipment.

See the `nidaqmx` examples for how to use `NI-DAQ`.

The returned object from the `connect()` method is equivalent to importing the `NI-DAQ` package.

For example:

```
nidaqmx = record.connect()
with nidaqmx.Task() as task:
    task.ai_channels.add_ai_voltage_chan('Dev1/ai0')
    voltage = task.read()
```

is equivalent to:

```
import nidaqmx
with nidaqmx.Task() as task:
    task.ai_channels.add_ai_voltage_chan('Dev1/ai0')
    voltage = task.read()
```

The *backend* value must be equal to *NIDAQ* 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 NIDAQ.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**property constants**

Returns the *nidaqmx.constants* module.

**property CtrFreq**

Returns the *CtrFreq* class.

**property CtrTick**

Returns the *CtrTick* class.

**property CtrTime**

Returns the *CtrTime* class.

**property DaqError**

Returns the *DaqError* class.

**property DaqResourceWarning**

Returns the *DaqResourceWarning* class.

**property DaqWarning**

Returns the *DaqWarning* class.

**property errors**

Returns the *nidaqmx.errors* module.

**property Scale**

Returns the *Scale* class.

**property stream\_readers**

Returns the *nidaqmx.stream\_readers* module.

**property stream\_writers**

Returns the *nidaqmx.stream\_writers* module.

**property system**

Returns the *nidaqmx.system* module.

**property Task**

Returns the *Task* class.

**property types**

Returns the *nidaqmx.types* module.

**property utils**

Returns the *nidaqmx.utils* module.

**property version**

The NI-DAQmx driver version number.

Type  
str

## msl.equipment.connection\_prologix module

Uses [Prologix](#) hardware to establish a connection to the equipment.

**class** msl.equipment.connection\_prologix.**ConnectionPrologix**(*record*)

Bases: [Connection](#)

Uses [Prologix](#) hardware to establish a connection to the equipment.

For the GPIB-ETHERNET Controller, the format of the [address](#) is `Prologix::HOST::1234::PAD[::SAD]`, where PAD (Primary Address) is a decimal value between 0 and 30 and SAD (Secondary Address) is a decimal value between 96 and 126. SAD is optional. For example, `Prologix::192.168.1.110::1234::6` or `Prologix::192.168.1.110::1234::6::96`.

For the GPIB-USB Controller, the format of the [address](#) is `Prologix::PORT::PAD[::SAD]`, where PAD (Primary Address) is a decimal value between 0 and 30 and SAD (Secondary Address) is a decimal value between 96 and 126. SAD is optional. For example, `Prologix::COM3::6` or `Prologix::/dev/ttyUSB0::6::112`.

The [properties](#) for a [Prologix](#) connection supports the following key-value pairs in the [Connections Database](#) and any of the key-value pairs supported by [ConnectionSerial](#) or [ConnectionSocket](#) (depending on whether a GPIB-USB or a GPIB-ETHERNET Controller is used):

```
'eoi': int, 0 or 1
'eos': int, 0, 1, 2 or 3
'eot_char': int, an ASCII value less than 256
'eot_enable': int, 0 or 1
'mode': int, 0 or 1 [default: 1]
'read_tmo_ms': int, a timeout value between 1 and 3000 milliseconds
```

The [backend](#) value must be equal to [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 [connect\(\)](#) method to connect to the equipment.

### Parameters

**record** ([EquipmentRecord](#)) – A record from an [Equipment-Register Database](#).

**controllers** = {}

A [dict](#) of all [Prologix](#) Controllers that are being used to communicate with GPIB devices.

**selected\_addresses** = {}

A [dict](#) of the currently-selected GPIB address for all [Prologix](#) Controllers.

### property encoding

The encoding that is used for [read\(\)](#) and [write\(\)](#) operations.

Type  
str

**property encoding\_errors**

The error handling scheme to use when encoding and decoding messages.

For example: *strict*, *ignore*, *replace*, *xmlcharrefreplace*, *backslashreplace*

**Type**

`str`

**property read\_termination**

The termination character sequence that is used for the `read()` method.

Reading stops when the equipment stops sending data or the `read_termination` character sequence is detected. If you set the `read_termination` to be equal to a variable of type `str` it will automatically be encoded.

**Type**

`bytes` or `None`

**property write\_termination**

The termination character sequence that is appended to `write()` messages.

If you set the `write_termination` to be equal to a variable of type `str` it will automatically be encoded.

**Type**

`bytes` or `None`

**property max\_read\_size**

The maximum number of bytes that can be `read()`.

**Type**

`int`

**property timeout**

The timeout, in seconds, for `read()` and `write()` operations.

**Type**

`float` or `None`

**property rstrip**

Whether to remove trailing whitespace from `read()` messages.

**Type**

`bool`

**property controller**

`ConnectionSerial` or `ConnectionSocket`: The connection to the Prologix Controller for this equipment.

Depends on whether a GPIB-USB or a GPIB-ETHERNET Controller is being used to communicate with the equipment.

**disconnect()**

Calling this method does not close the underlying `ConnectionSerial` or `ConnectionSocket` connection to the Prologix Controller since the connection to the Prologix Controller may still be required to send messages to other devices via GPIB.

Calling this method sets the `controller` to be `None`.



**group\_execute\_trigger(\*addresses)**

Send the Group Execute Trigger command to equipment at the specified addresses.

Up to 15 addresses may be specified. If no address is specified then the Group Execute Trigger command is issued to the currently-addressed equipment.

**Parameters**

**addresses** – The primary (and optional secondary) GPIB addresses. If a secondary address is specified then it must follow its corresponding primary address. For example:

- `group_execute_trigger(1, 11, 17)` → primary, primary, primary
- `group_execute_trigger(3, 96, 12, 21)` → primary, secondary, primary, primary

**Returns**

`int` – The number of bytes written.

**read(\*\*kwargs)**

Read a message from the equipment.

**Parameters**

**\*\*kwargs** – All keyword arguments are passed to `read()`.

**Returns**

`str`, `bytes` or `ndarray` – The message from the equipment. If `dtype` is specified, then the message is returned as an `ndarray`, if `decode` is `True` then the message is returned as a `str`, otherwise the message is returned as `bytes`.

**write(message, \*\*kwargs)**

Write a message to the equipment.

**Parameters**

- **message** (`str` or `bytes`) – The message to write to the equipment.
- **\*\*kwargs** – All keyword arguments are passed to `write()`.

**Returns**

`int` – The number of bytes written.

**query(message, \*\*kwargs)**

Convenience method for performing a `write()` followed by a `read()`.

**Parameters**

- **message** (`str` or `bytes`) – The message to write to the equipment.
- **\*\*kwargs** – All keyword arguments are passed to `query()`.

**Returns**

`str`, `bytes` or `ndarray` – The message from the equipment. If `dtype` is specified, then the message is returned as an `ndarray`, if `decode` is `True` then the message is returned as a `str`, otherwise the message is returned as `bytes`.

**property query\_auto**

Whether to send ++auto 1 before and ++auto 0 after a `query()` to the Prologix Controller.

**Type**`bool`**version()**

Get the version of the `Prologix` Controller.

**Returns**

`str` – The type of the Controller (GPIO-USB or GPIO-ETHERNET) and the version of the firmware.

**static parse\_address(address)**

Parse the address to determine the connection class and the GPIO address.

**Parameters**

**address** (`str`) – The address of a `ConnectionRecord`.

**Returns**

`dict` or `None` – If *address* is valid for a Prologix connection then the key-value pairs are:

- **class**, `ConnectionSocket` or `ConnectionSerial`  
The underlying connection class to use (not instantiated).
- **name**, `str`  
The name of the connection class.
- **pad**, `int`  
The primary GPIO address.
- **sad**, `int` or `None`  
The secondary GPIO address.

otherwise `None` is returned.

`msl.equipment.connection_prologix.find_prologix(hosts=None, timeout=1)`

Find all Prologix ENET-GPIO devices that are on the network.

To resolve the MAC address of a Prologix device, the `arp` program must be installed. On Linux, install `net-tools`. On Windows and macOS, `arp` should already be installed.

**Parameters**

- **hosts** (`list` of `str`, optional) – The IP address(es) on the computer to use to look for Prologix ENET-GPIO devices. If not specified, then use all network interfaces.
- **timeout** (`float`, optional) – The maximum number of seconds to wait for a reply.

**Returns**

`dict` – The information about the Prologix ENET-GPIO devices that were found.

## msl.equipment.connection\_pyvisa module

Uses [PyVISA](#) as the backend to communicate with the equipment.

**class** msl.equipment.connection\_pyvisa.**ConnectionPyVISA**(*record*)

Bases: [Connection](#)

Uses [PyVISA](#) to establish a connection to the equipment.

The *backend* value must be equal to [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](#).

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

### Parameters

**record** ([EquipmentRecord](#)) – A record from an *Equipment-Register Database*.

### property resource

The [PyVISA](#) resource that is used for the connection.

This is the [Resource](#) that would have been returned if you did the following in a script:

```
import pyvisa
rm = pyvisa.ResourceManager()
resource = rm.open_resource('ASRL3::INSTR')
```

### Type

[Resource](#)

### disconnect()

Calls [close\(\)](#).

**static resource\_manager**(*visa\_library=None*)

Return the [PyVISA ResourceManager](#).

### Parameters

**visa\_library** ([VisaLibraryBase](#) or [str](#), optional) – The library to use for [PyVISA](#). For example:

- @ivi to use [IVI](#)
- @ni to use [NI-VISA](#) (only supported in [PyVISA](#) <1.11)
- @py to use [PyVISA-py](#)
- @sim to use [PyVISA-sim](#)

If *None* then [PyVISA\\_LIBRARY](#) will be used.

### Returns

[ResourceManager](#) – The [PyVISA](#) Resource Manager.

### Raises

- **ValueError** – If the [PyVISA](#) backend wrapper cannot be found.
- **OSError** – If an [IVI](#) library cannot be found.

**static resource\_class(record)**

Get the [PyVISA Resource class](#).

**Parameters**

**record** ([EquipmentRecord](#) or [ConnectionRecord](#)) – An equipment or connection record from a [Database](#).

**Returns**

A [Resource](#) subclass – The [PyVISA Resource class](#) that can open the *record*.

## **msl.equipment.connection\_sdk module**

Base class for equipment that use the SDK provided by the manufacturer for the connection.

**class** `msl.equipment.connection_sdk.ConnectionSDK(record, libtype, path=None)`

Bases: [Connection](#)

Base class for equipment that use the SDK provided by the manufacturer for the connection.

The [backend](#) value must be equal to [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 [connect\(\)](#) method to connect to the equipment.

**Parameters**

- **record** ([EquipmentRecord](#)) – A record from an [Equipment-Register Database](#).
- **libtype** ([str](#)) – The library type. See [LoadLibrary](#) for more information.
- **path** ([str](#), optional) – The path to the SDK (if *record.connection.address* does not contain this information).

**Raises**

- **OSError** – If the shared library cannot be loaded.
- **TypeError** – If either *record* or *libtype* is invalid.

**property assembly**

The reference to the .NET assembly.

**Type**

[assembly](#)

**property gateway**

The reference to the JAVA gateway.

**Type**

[gateway](#)

**property path**

The path to the SDK file.

**Type**

[str](#)

**property sdk**

The reference to the SDK object.

**Type**

`lib`

**log\_errcheck**(*result, func, arguments*)

Convenience method for logging an `errcheck`

**static parse\_address**(*address*)

Get the file path from an address.

**Parameters**

**address** (`str`) – The address of a `ConnectionRecord`.

**Returns**

`dict` or `None` – The file path or `None` if *address* is not valid for an SDK.

**msl.equipment.connection\_serial module**

Base class for equipment that is connected through a serial port.

**class** `msl.equipment.connection_serial.ConnectionSerial`(*record*)

Bases: `ConnectionMessageBased`

Base class for equipment that is connected through a serial port.

The *properties* for a serial connection supports the following key-value pairs in the `Connections Database` (see also `serial.Serial` for more details about each parameter):

```
'baud_rate': int, the baud rate [default: 9600]
'data_bits': int, the number of data bits, e.g. 5, 6, 7, 8 [default: 8]
'dsr_dtr': bool, enable hardware (DSR/DTR) flow control [default: False]
'encoding': str, the encoding to use [default: 'utf-8']
'encoding_errors': str, encoding error handling scheme, e.g. 'strict',
↳ 'ignore' [default: 'strict']
'inter_byte_timeout': float or None, the inter-character timeout
↳ [default: None]
'max_read_size': int, the maximum number of bytes that can be read
↳ [default: 1 MB]
'parity': str or None, parity checking, e.g. 'even', 'odd' [default: None]
'read_termination': str or None, read until this termination sequence is
↳ found [default: '\n']
'rstrip': bool, whether to remove trailing whitespace from "read"
↳ messages [default: False]
'rts_cts': bool, enable hardware (RTS/CTS) flow control [default: False]
'stop_bits': int or float, the number of stop bits, e.g. 1, 1.5, 2
↳ [default: 1]
'termination': shortcut for setting both 'read_termination' and 'write_
↳ termination' to this value
'timeout': float or None, the timeout (in seconds) for read and write
↳ operations [default: None]
'write_termination': str or None, termination sequence appended to write
```

(continues on next page)

(continued from previous page)

```
↪ messages [default: '\r\n']  
'xon_xoff': bool, enable software flow control [default: False]
```

The *backend* value must be equal to *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 *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**Raises**

*MSLConnectionError* – If the serial port cannot be opened.

**property serial**

The reference to the serial object.

**Type**

*serial.Serial*

**property baud\_rate**

The baud rate setting.

**Type**

*int*

**property data\_bits**

The number of data bits.

**Type**

*DataBits*

**property stop\_bits**

The stop bit setting.

**Type**

*StopBits*

**property parity**

The parity setting.

**Type**

*Parity*

**disconnect()**

Close the serial port.

**static parse\_address(address)**

Get the serial port from an address.

**Parameters**

**address** (*str*) – The address of a *ConnectionRecord*

**Returns**

*dict* or *None* – The serial port in a format that is valid for PySerial (i.e., ASRL3 becomes COM3) or *None* if the port cannot be determined from *address*.

## msl.equipment.connection\_socket module

Base classes for equipment that is connected through a socket.

**class** msl.equipment.connection\_socket.ConnectionSocket(*record*)

Bases: *ConnectionMessageBased*

Base class for equipment that is connected through a socket.

The *properties* for a socket connection supports the following key-value pairs in the *Connections Database* (see also *socket* for more details):

```

'buffer_size': int, the maximum number of bytes to read at a time
↳ [default: 4096]
'encoding': str, the encoding to use [default: 'utf-8']
'encoding_errors': str, encoding error handling scheme, e.g. 'strict',
↳ 'ignore' [default: 'strict']
'family': str, the address family, e.g., 'INET', 'INET6', 'IPX' [default:
↳ 'INET']
'max_read_size': int, the maximum number of bytes that can be read
↳ [default: 1 MB]
'proto': int, the socket protocol number [default: 0]
'read_termination': str or None, read until this termination sequence is
↳ found [default: '\n']
'rstrip': bool, whether to remove trailing whitespace from "read"
↳ messages [default: False]
'socket_type': str, the socket type, e.g. 'STREAM', 'DGRAM' [default:
↳ 'STREAM']
'termination': shortcut for setting both 'read_termination' and 'write_
↳ termination' to this value
'timeout': float or None, the timeout (in seconds) for read and write
↳ operations [default: None]
'write_termination': str or None, termination sequence appended to write
↳ messages [default: '\r\n']

```

The *backend* value must be equal to *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 *connect()* method to connect to the equipment.

### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

### Raises

*MSLConnectionError* – If the socket cannot be opened.

### property byte\_buffer

Returns the reference to the byte buffer.

### Type

*bytearray*

### property host

The host (IP address).

**Type**

`str`

**property port**

The port number.

**Type**

`int`

**property socket**

The reference to the socket.

**Type**

`socket`

**static parse\_address(address)**

Get the host and port from an address.

**Parameters**

**address** (`str`) – The address of a *ConnectionRecord*.

**Returns**

`dict` or `None` – The value of the host and the port or `None` if *address* is not valid for a socket.

**disconnect()**

Close the socket.

**reconnect(max\_attempts=1)**

Reconnect to the equipment.

**Parameters**

**max\_attempts** (`int`, optional) – The maximum number of attempts to try to reconnect with the equipment. If `< 1` or `None` then keep trying until a connection is successful. If the maximum number of attempts has been reached then an exception is raise.

## **msl.equipment.connection\_tcpip\_hislip module**

Base class for equipment that use the HiSLIP communication protocol.

**class** `msl.equipment.connection_tcpip_hislip.ConnectionTCPIPHiSLIP(record)`

Bases: *ConnectionMessageBased*

Base class for equipment that use the HiSLIP communication protocol.

The *properties* for a HiSLIP connection supports the following key-value pairs in the *Connections Database*:

```
'buffer_size': int, the maximum number of bytes to read at a time,
↳ [default: 4096]
'encoding': str, the encoding to use [default: 'utf-8']
'encoding_errors': str, encoding error handling scheme, e.g. 'strict',
↳ 'ignore' [default: 'strict']
'lock_timeout': float or None, the timeout (in seconds) to wait for a
```

(continues on next page)



(continued from previous page)

```

↪ lock [default: 0]
'↪ max_read_size': int, the maximum number of bytes that can be read
↪ [default: 1 MB]
'↪ rstrip': bool, whether to remove trailing whitespace from "read"
↪ messages [default: False]
'↪ timeout': float or None, the timeout (in seconds) for read and write
↪ operations [default: None]

```

The *backend* value must be equal to *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 *connect()* method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

#### property host

The host (IP address).

#### Type

*str*

#### property port

The port number.

#### Type

*int*

#### property asynchronous

The reference to the asynchronous client.

#### Type

*AsyncClient*

#### property synchronous

The reference to the synchronous client.

#### Type

*SyncClient*

#### static parse\_address(address)

Parse the address for valid TCPIP HiSLIP fields.

#### Parameters

**address** (*str*) – The address of a *ConnectionRecord*.

#### Returns

*dict* or *None* – The board number, hostname, LAN device name, and HiSLIP port number of the device or *None* if *address* is not valid for a TCPIP HiSLIP connection.

#### property max\_read\_size

The maximum number of bytes that can be *read()*.

#### Type

*int*

**property lock\_timeout**

The time, in seconds, to wait to acquire a lock.

**Type**

`float`

**disconnect()**

Close the connection to the HiSLIP server.

**reconnect(*max\_attempts=1*)**

Reconnect to the equipment.

**Parameters**

**max\_attempts** (`int`, optional) – The maximum number of attempts to try to reconnect with the equipment. If `< 1` or `None` then keep trying until a connection is successful. If the maximum number of attempts has been reached then an exception is raised.

**read\_stb()**

Read the status byte from the device.

**Returns**

`int` – The status byte.

**trigger()**

Send the trigger message (emulates a GPIB Group Execute Trigger event).

**clear()**

Send the *clear* command to the device.

**lock(*lock\_string=""*)**

Acquire the device's lock.

**Parameters**

**lock\_string** (`str`, optional) – An ASCII string that identifies this lock. If not specified, then an exclusive lock is requested, otherwise the string indicates an identification of a shared-lock request.

**Returns**

`bool` – Whether acquiring the lock was successful.

**unlock()**

Release the lock acquired by `lock()`.

**Returns**

`bool` – Whether releasing the lock was successful.

**lock\_status()**

Request the lock status from the HiSLIP server.

**Returns**

- `bool` – Whether the HiSLIP server has an exclusive lock with a client.
- `int` – The number of HiSLIP clients that have a lock with the HiSLIP server.

**remote\_local\_control(*request*)**

Send a GPIB-like remote/local control request.

**Parameters**

**request** (*int*) – The request to perform.

- 0 – Disable remote, *VI\_GPIB\_REN\_DEASSERT*
- 1 – Enable remote, *VI\_GPIB\_REN\_ASSERT*
- 2 – Disable remote and go to local, *VI\_GPIB\_REN\_DEASSERT\_GTL*
- 3 – Enable Remote and go to remote, *VI\_GPIB\_REN\_ASSERT\_ADDRESS*
- 4 – Enable remote and lock out local, *VI\_GPIB\_REN\_ASSERT\_LLO*
- 5 – Enable remote, go to remote, and set local lockout, *VI\_GPIB\_REN\_ASSERT\_ADDRESS\_LLO*
- 6 – go to local without changing REN or lockout state, *VI\_GPIB\_REN\_ADDRESS\_GTL*

**msl.equipment.connection\_tcpip\_vxi11 module**

Base class for equipment that use the VXI-11 communication protocol.

**class** msl.equipment.connection\_tcpip\_vxi11.**ConnectionTCPIP VXI11**(*record*)

Bases: *ConnectionMessageBased*

Base class for equipment that use the VXI-11 communication protocol.

The *properties* for a VXI-11 connection supports the following key-value pairs in the *Connections Database*:

```
'buffer_size': int, the maximum number of bytes to read at a time
↳ [default: 4096]
'encoding': str, the encoding to use [default: 'utf-8']
'encoding_errors': str, encoding error handling scheme, e.g. 'strict',
↳ 'ignore' [default: 'strict']
'lock_timeout': float or None, the timeout (in seconds) to wait for a
↳ lock [default: 0]
'max_read_size': int, the maximum number of bytes that can be read
↳ [default: 1 MB]
'port': int, the port to use instead of calling the RPC Port Mapper
↳ function [default: None]
'read_termination': str or None, read until this termination character is
↳ found [default: None]
'rstrip': bool, whether to remove trailing whitespace from "read"
↳ messages [default: False]
'termination': alias for 'read_termination'
'timeout': float or None, the timeout (in seconds) for read and write
↳ operations [default: None]
```

The *backend* value must be equal to *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 *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**property byte\_buffer**

Returns the reference to the byte buffer.

**Type**

`bytearray`

**property host**

The host (IP address).

**Type**

`str`

**property port**

The port number.

**Type**

`int`

**property socket**

The reference to the socket.

**Type**

`socket`

**static parse\_address(address)**

Parse the address for valid TCPIP VXI-11 fields.

**Parameters**

**address** (`str`) – The address of a *ConnectionRecord*.

**Returns**

`dict` or `None` – The board number, hostname, and LAN device name of the device or `None` if *address* is not valid for a TCPIP VXI-11 connection.

**disconnect()**

Unlink and close the sockets.

**reconnect(max\_attempts=1)**

Reconnect to the equipment.

**Parameters**

**max\_attempts** (`int`, optional) – The maximum number of attempts to try to reconnect with the equipment. If `< 1` or `None` then keep trying until a connection is successful. If the maximum number of attempts has been reached then an exception is raised.

**property lock\_timeout**

The time, in seconds, to wait to acquire a lock.

**Type**

`float`

**abort()**

Stop an in-progress request.

**read\_stb()**

Read the status byte from the device.

**Returns**

**int** – The status byte.

**trigger()**

Send a trigger to the device.

**clear()**

Send the *clear* command to the device.

**remote()**

Place the device in a remote state wherein all programmable local controls are disabled.

**local()**

Place the device in a local state wherein all programmable local controls are enabled.

**lock()**

Acquire the device's lock.

**unlock()**

Release the lock acquired by *lock()*

**enable\_sqr(enable, handle)**

Enable or disable the sending of *device\_intr\_srq* RPCs by the network instrument server.

**Parameters**

- **enable** (**bool**) – Whether to enable or disable interrupts.
- **handle** (**bytes**) – Host specific data (maximum length is 40 characters).

**docmd(cmd, value, fmt)**

Allows for a variety of commands to be executed.

**Parameters**

- **cmd** (**int**) – An IEEE 488 command messages, (e.g., to send a group execute trigger, GET, command the value of *cmd* would be 0x08).
- **value** (**bool**, **int** or **float**) – The value to use with *cmd*.
- **fmt** (**str**) – How to format *value*. See [Format Characters](#) for more details. Do not include the byte-order character. Network (big-endian) order will always be used.

**Returns**

**bytes** – The results defined by *cmd*.

**destroy\_link()**

Destroy the link with the device.

**create\_intr\_chan(host\_addr, host\_port, prog\_num, prog\_vers, prog\_family)**

Inform the network instrument server to establish an interrupt channel.

**Parameters**

- **host\_addr** (**int**) – Host servicing the interrupt.

- **host\_port** (*int*) – Valid port number on the client.
- **prog\_num** (*int*) – Program number.
- **prog\_vers** (*int*) – Program version number.
- **prog\_family** (*int*) – The underlying socket protocol family type (IPPROTO\_TCP or IPPROTO\_UDP).

**destroy\_intr\_chan()**

Inform the network instrument server to close its interrupt channel.

**msl.equipment.connection\_zeromq module**

Base class for equipment that use the *ZeroMQ* communication protocol.

**class** msl.equipment.connection\_zeromq.**ConnectionZeroMQ**(*record*)

Bases: *ConnectionMessageBased*

Base class for equipment that use the *ZeroMQ* communication protocol.

The *properties* for a ZeroMQ connection supports the following key-value pairs in the *Connections Database*:

```
'encoding': str, the encoding to use [default: 'utf-8']
'encoding_errors': str, encoding error handling scheme, e.g. 'strict',
↳ 'ignore' [default: 'strict']
'max_read_size': int, the maximum number of bytes that can be read
↳ [default: 1 MB]
'protocol': str, the ZeroMQ protocol [default: 'tcp']
'rstrip': bool, whether to remove trailing whitespace from "read"
↳ messages [default: False]
'socket_type': str, the ZeroMQ socket type [default: 'REQ']
'timeout': float or None, the timeout (in seconds) for read and write
↳ operations [default: None]
```

The *backend* value must be equal to *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 *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**Raises**

*MSLConnectionError* – If the socket cannot be opened.

**property context**

Reference to the ZeroMQ context.

**Type**

*Context*

**disconnect()**

Close the connection.

**property host**

The host (IP address).

**Type**

`str`

**property max\_read\_size**

The maximum number of bytes that can be `read()`.

**Type**

`int`

**static parse\_address(address)**

Parse the address for valid ZeroMQ fields.

**Parameters**

**address** (`str`) – The address of a `ConnectionRecord`.

**Returns**

`dict` or `None` – The host and port number of the device or `None` if `address` is not valid for a ZeroMQ connection.

**property port**

The port number.

**Type**

`int`

**reconnect(max\_attempts=1)**

Reconnect to the equipment.

**Parameters**

**max\_attempts** (`int`, optional) – The maximum number of attempts to try to reconnect with the equipment. If `< 1` or `None` then keep trying until a connection is successful. If the maximum number of attempts has been reached then an exception is raised.

**property socket**

Reference to the ZeroMQ socket.

**Type**

`Socket`

## **msl.equipment.constants module**

MSL-Equipment constants.

```
class msl.equipment.constants.Backend(value, names=None, *, module=None,
                                     qualname=None, type=None, start=1,
                                     boundary=None)
```

Bases: `IntEnum`

The software backend to use for the communication system.

**UNKNOWN = 0**

**MSL** = 1

**PyVISA** = 2

**NIDAQ** = 3

```
class msl.equipment.constants.Interface(value, names=None, *, module=None,
                                         qualname=None, type=None, start=1,
                                         boundary=None)
```

Bases: [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 backend.

**NONE** = 0

**SDK** = 1

**SERIAL** = 2

**SOCKET** = 3

**PROLOGIX** = 4

**TCPIP\_VXI11** = 5

**TCPIP\_HISLIP** = 6

**ZMQ** = 7

```
class msl.equipment.constants.Parity(value, names=None, *, module=None,
                                         qualname=None, type=None, start=1,
                                         boundary=None)
```

Bases: [Enum](#)

The parity type to use for Serial communication.

**NONE** = 'N'

**ODD** = 'O'

**EVEN** = 'E'

**MARK** = 'M'

**SPACE** = 'S'

```
class msl.equipment.constants.StopBits(value, names=None, *, module=None,
                                         qualname=None, type=None, start=1,
                                         boundary=None)
```

Bases: [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(value, names=None, *, module=None,
                                       qualname=None, type=None, start=1,
                                       boundary=None)
```

Bases: [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 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](#).

This class should be accessed through the [database\(\)](#) method after a [Config](#) object has been created.

### Parameters

**path** ([str](#)) – The path to an XML [Configuration File](#).

### Raises

- **OSError** – If *path* does not exist or if the [Configuration File](#) is invalid.
- **AttributeError** – If an <equipment> XML tag is specified in the [Configuration File](#) and it does not uniquely identify an equipment record in an [Equipment-Register Database](#).
- **ValueError** – If an [alias](#) has been specified multiple times for the same [EquipmentRecord](#) or if the name of the Sheet in an Excel spreadsheet is invalid.

### property equipment

[EquipmentRecord](#)'s that were listed as <equipment> XML tags in the [Configuration File](#).

### Type

[dict](#)

### property path

The path to the [Configuration File](#).

### Type

[str](#)

**connections(\*\*kwargs)**

Search the *Connections Database* to find all *ConnectionRecord*'s that match the specified criteria.

**Parameters**

**\*\*kwargs** – The argument names can be any of the *ConnectionRecord* property names or a `flags` argument of type `int` for performing the search, see `re.search()`. For testing regex expressions online you can use [this](#) website.

If a *kwarg* is `properties` then the value must be a `dict`. See the examples below.

**Examples**

- `connections()` → a list of all *ConnectionRecord*'s
- `connections(manufacturer='Keysight')` → a list of all *ConnectionRecord*'s that have Keysight as the manufacturer
- `connections(manufacturer='Agilent|Keysight')` → a list of all *ConnectionRecord*'s that are from Agilent or Keysight
- `connections(manufacturer='H.*P')` → a list of all *ConnectionRecord*'s that have Hewlett Packard (or HP) as the manufacturer
- `connections(manufacturer='Agilent', model='^34')` → a list of all *ConnectionRecord*'s that have Agilent as the manufacturer and a model number beginning with '34'
- `connections(interface=Interface.SERIAL)` → a list of all *ConnectionRecord*'s that use SERIAL for the connection bus
- `connections(interface='SDK')` → a list of all *ConnectionRecord*'s that use the manufacturers SDK to control the equipment
- `connections(backend=Backend.PyVISA)` → a list of all *ConnectionRecord*'s that use PyVISA as the backend
- `connections(backend='MSL')` → a list of all *ConnectionRecord*'s that use MSL as the backend
- `connections(properties={'baud_rate': 115200})` → a list of all *ConnectionRecord*'s that specify a baud rate equal to 115200 in the Properties field

**Returns**

`list` of *ConnectionRecord* – The connection records that match the search criteria.

**Raises**

**NameError** – If the name of an input argument is not a *ConnectionRecord* property name or `flags`.

**records(\*\*kwargs)**

Search the *Equipment-Register Database* to find all *EquipmentRecord*'s that match the specified criteria.

**Parameters**

**\*\*kwargs** – The argument names can be any of the *EquipmentRecord* property names or a **flags** argument of type `int` for performing the search, see `re.search()`. For testing regex expressions online you can use [this](#) website.

If a *kwarg* is `connection` then the value will be used to test which *EquipmentRecord*'s have a `connection` value that is either `None` or *ConnectionRecord*. See the examples below.

**Examples**

- `records()` → a list of all *EquipmentRecord*'s
- `records(manufacturer='Agilent')` → a list of all *EquipmentRecord*'s that are from Agilent
- `records(manufacturer='Agilent|Keysight')` → a list of all *EquipmentRecord*'s that are from Agilent or Keysight
- `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(manufacturer=r'H.*P')` → a list of all *EquipmentRecord*'s that have Hewlett Packard (or HP) as the manufacturer
- `records(description='I-V Converter')` → a list of all *EquipmentRecords* that contain 'I-V Converter' in the description field
- `records(connection=True)` → a list of all *EquipmentRecords* that can be connected to

**Returns**

`list` of *EquipmentRecord* – The equipment records that match the search criteria.

**Raises**

**NameError** – If the name of an input argument is not an *EquipmentRecord* property name or **flags**.

**msl.equipment.dns\_service\_discovery module**

Implementation of the Multicast DNS and DNS-Based Service Discovery protocols.

## References

- [RFC-1035](#) – *Domain Names - Implementation and Specification*, **ISI**, November 1987.
- [RFC-6762](#) – *Multicast DNS*, **Apple Inc.**, February 2013.
- [RFC-6763](#) – *DNS-Based Service Discovery*, **Apple Inc.**, February 2013.

`msl.equipment.dns_service_discovery.find_lxi(hosts=None, timeout=1)`

Find all LXI devices that support the mDNS and DNS Service Discovery protocols.

### Parameters

- **hosts** ([list](#) of [str](#), optional) – The IP address(es) on the computer to use to broadcast the message. If not specified, then broadcast on all network interfaces.
- **timeout** ([float](#), optional) – The maximum number of seconds to wait for a reply.

### Returns

[dict](#) – The information about the HiSLIP, VXI-11 and SCPI-RAW devices that were found.

## msl.equipment.exceptions module

Exceptions used by MSL-Equipment.

**exception** `msl.equipment.exceptions.MSLConnectionError`

Bases: [OSError](#)

Base class for all MSL [Connection](#) exceptions.

**exception** `msl.equipment.exceptions.MSLTimeoutError`

Bases: [MSLConnectionError](#)

A timeout exception for I/O operations.

**exception** `msl.equipment.exceptions.ResourceClassNotFound(record)`

Bases: [MSLConnectionError](#)

Exception if a resource class cannot be found to connect to the equipment.

**exception** `msl.equipment.exceptions.AimTTiError`

Bases: [MSLConnectionError](#)

Exception for equipment from Aim and Thurlby Thandar Instruments.

**exception** `msl.equipment.exceptions.AvantesError`

Bases: [MSLConnectionError](#)

Exception for equipment from Avantes.

**exception** `msl.equipment.exceptions.BenthamError`

Bases: [MSLConnectionError](#)

Exception for equipment from Bentham.

**exception** `msl.equipment.exceptions.CMIErrors`

Bases: *MSLConnectionError*

Exception for equipment from the Czech Metrology Institute.

**exception** `msl.equipment.exceptions.DataRayError`

Bases: *MSLConnectionError*

Exception for equipment from DataRay Inc.

**exception** `msl.equipment.exceptions.EnergetiqError`

Bases: *MSLConnectionError*

Exception for equipment from Energetiq.

**exception** `msl.equipment.exceptions.MKSInstrumentsError`

Bases: *MSLConnectionError*

Exception for equipment from MKS Instruments.

**exception** `msl.equipment.exceptions.NKTErrors`

Bases: *MSLConnectionError*

Exception for equipment from NKT Photonics.

**exception** `msl.equipment.exceptions.OmegaError`

Bases: *MSLConnectionError*

Exception for equipment from OMEGA.

**exception** `msl.equipment.exceptions.OptoSigmaError`

Bases: *MSLConnectionError*

Exception for equipment from OptoSigma.

**exception** `msl.equipment.exceptions.OptronicLaboratoriesError`

Bases: *MSLConnectionError*

Exception for equipment from Optronic Laboratories.

**exception** `msl.equipment.exceptions.PicoTechError`

Bases: *MSLConnectionError*

Exception for equipment from Pico Technology.

**exception** `msl.equipment.exceptions.PrincetonInstrumentsError`

Bases: *MSLConnectionError*

Exception for equipment from Princeton Instruments.

**exception** `msl.equipment.exceptions.RaicolCrystalsError`

Bases: *MSLConnectionError*

Exception for equipment from Raicol Crystals.

**exception** `msl.equipment.exceptions.ThorlabsError`

Bases: *MSLConnectionError*

Exception for equipment from Thorlabs.

## msl.equipment.factory module

Establish a connection to the equipment.

`msl.equipment.factory.connect(record, demo=None)`

Factory function to establish a connection to the equipment.

### Parameters

- **record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.
- **demo** (*bool*, optional) – Whether to simulate a connection to the equipment by opening a connection in demo mode. This allows you to test your code if the equipment is not physically connected to a computer.

If *None* then the *demo* value is determined from the *DEMO\_MODE* attribute.

### Returns

A *Connection* subclass.

`msl.equipment.factory.find_interface(address)`

Find the interface enum.

### Parameters

**address** (*str*) – The address of a *ConnectionRecord*.

### Returns

*constants.Interface* – The interface to use for *address*.

## msl.equipment.hislip module

Implementation of the *HiSLIP* protocol for a client.

This module implements the following IVI Protocol Specification:

*IVI-6.1: High-Speed LAN Instrument Protocol (HiSLIP) v2.0 April 23, 2020*

```
class msl.equipment.hislip.MessageType(value, names=None, *, module=None,
                                       qualname=None, type=None, start=1,
                                       boundary=None)
```

Bases: *IntEnum*

Message types.

**Initialize** = 0

**InitializeResponse** = 1

**FatalError** = 2

**Error** = 3

**AsyncLock** = 4

**AsyncLockResponse** = 5

**Data** = 6

DataEnd = 7  
DeviceClearComplete = 8  
DeviceClearAcknowledge = 9  
AsyncRemoteLocalControl = 10  
AsyncRemoteLocalResponse = 11  
Trigger = 12  
Interrupted = 13  
AsyncInterrupted = 14  
AsyncMaximumMessageSize = 15  
AsyncMaximumMessageSizeResponse = 16  
AsyncInitialize = 17  
AsyncInitializeResponse = 18  
AsyncDeviceClear = 19  
AsyncServiceRequest = 20  
AsyncStatusQuery = 21  
AsyncStatusResponse = 22  
AsyncDeviceClearAcknowledge = 23  
AsyncLockInfo = 24  
AsyncLockInfoResponse = 25  
GetDescriptors = 26  
GetDescriptorsResponse = 27  
StartTLS = 28  
AsyncStartTLS = 29  
AsyncStartTLSResponse = 30  
EndTLS = 31  
AsyncEndTLS = 32  
AsyncEndTLSResponse = 33  
GetSaslMechanismList = 34  
GetSaslMechanismListResponse = 35  
AuthenticationStart = 36

**AuthenticationExchange** = 37

**AuthenticationResult** = 38

**class** msl.equipment.hislip.**ErrorType**(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: [IntEnum](#)

Error types.

**UNIDENTIFIED** = 0

**BAD\_HEADER** = 1

**CHANNELS\_INACTIVATED** = 2

**INVALID\_INIT\_SEQUENCE** = 3

**MAX\_CLIENTS** = 4

**BAD\_MESSAGE\_TYPE** = 1

**BAD\_CONTROL\_CODE** = 2

**BAD\_VENDOR** = 3

**MESSAGE\_TOO\_LARGE** = 4

**AUTHENTICATION\_FAILED** = 5

**exception** msl.equipment.hislip.**HiSLIPException**(*message\_type, control\_code, reason=None*)

Bases: [Exception](#)

Base class for HiSLIP exceptions.

#### Parameters

- **message\_type** ([MessageType](#)) – The message type.
- **control\_code** ([int](#)) – The control code from the server response.
- **reason** ([str](#)) – Additional information to display in exception string.

#### property message

The error message that can be written to the server.

#### Type

[Message](#)

**exception** msl.equipment.hislip.**FatalError**(*control\_code, reason=None*)

Bases: [HiSLIPException](#)

Exception for a fatal error.

#### Parameters

- **control\_code** ([int](#)) – The control code from the server response.
- **reason** ([str](#)) – Additional information to display in exception string.



**exception** `msl.equipment.hislip.Error(control_code, reason=None)`

Bases: `HiSLIPException`

Exception for a non-fatal error.

#### Parameters

- **control\_code** (`int`) – The control code from the server response.
- **reason** (`str`) – Additional information to display in exception string.

**class** `msl.equipment.hislip.Message(control_code=0, parameter=0, payload=b'')`

Bases: `object`

Create a new HiSLIP message.

#### Parameters

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**header** = `<_struct.Struct object>`

**prologue** = `b'HS'`

**type** = `None`

**property** `length_payload`

The length of the payload.

#### Type

`int`

**pack**()

Convert the message to bytes.

#### Returns

`bytearray` – The messaged packed as bytes.

**static** `repack(unpack_fmt, pack_fmt, *args)`

Convert arguments from one byte format to another.

#### Parameters

- **unpack\_fmt** (`str`) – The format to convert the arguments to.
- **pack\_fmt** (`str`) – The format that the arguments are currently in.
- **\*args** – The arguments to convert.

#### Returns

`tuple` – The converted arguments.

**property** `size`

`int` The total size of the message.

```
class msl.equipment.hislip.FatalErrorMessage(control_code=0, parameter=0,
                                              payload=b")
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 2

```
class msl.equipment.hislip.ErrorMessage(control_code=0, parameter=0, payload=b")
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 3

```
class msl.equipment.hislip.Initialize(major, minor, client_id, sub_address)
```

Bases: [Message](#)

Create an Initialize message.

#### Parameters

- **major** ([int](#)) – The major version number of the HiSLIP protocol that the client supports.
- **minor** ([int](#)) – The minor version number of the HiSLIP protocol that the client supports.
- **client\_id** ([bytes](#)) – The vendor ID of the client. Must have a length of 2 characters.
- **sub\_address** ([bytes](#)) – A particular device managed by this server. For VISA clients this field corresponds to the VISA LAN device name (default is *hislip0*). The maximum length is 256 characters.

**type** = 0

```
class msl.equipment.hislip.InitializeResponse(control_code=0, parameter=0,
                                              payload=b")
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 1

#### **property encrypted**

Whether encryption is optional or mandatory.

##### Type

[bool](#)

#### **property initial\_encryption**

Whether the client shall switch to encrypted mode.

##### Type

[bool](#)

#### **property overlapped**

Whether the server is in overlapped or synchronous mode.

##### Type

[bool](#)

#### **property protocol\_version**

The (major, minor) version numbers of the HiSLIP protocol that the client and server are to use.

##### Type

[tuple](#)

#### **property session\_id**

The session ID.

##### Type

[int](#)

**class** `msl.equipment.hislip.Data`(*control\_code=0, parameter=0, payload=b''*)

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type = 6**

**class** msl.equipment.hislip.DataEnd(*control\_code=0, parameter=0, payload=b''*)

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type = 7**

**class** msl.equipment.hislip.AsyncLock(*control\_code=0, parameter=0, payload=b''*)

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type = 4**

**class** msl.equipment.hislip.AsyncLockResponse(*control\_code=0, parameter=0, payload=b''*)

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type = 5**

#### property error

Whether the request was an invalid attempt to release a lock that was not acquired or to request a lock already granted.

#### Type

*bool*

**property failed**

Whether a lock was requested but not granted (timeout expired).

**Type**

`bool`

**property success**

Whether requesting or releasing the lock was successful.

**Type**

`bool`

**property shared\_released**

Whether releasing a shared lock was successful.

**Type**

`bool`

**class** `msl.equipment.hislip.AsyncLockInfo`(*control\_code=0, parameter=0, payload=b''*)

Bases: `Message`

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 24

**class** `msl.equipment.hislip.AsyncLockInfoResponse`(*control\_code=0, parameter=0, payload=b''*)

Bases: `Message`

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 25

**property exclusive**

Whether the HiSLIP server has an exclusive lock with a client.

**Type**

`bool`

**property num\_locks**

The number of HiSLIP clients that have a lock with the HiSLIP server.

**Type**

`int`

```
class msl.equipment.hislip.AsyncRemoteLocalControl(control_code=0, parameter=0,
                                                    payload=b")
```

Bases: *Message*

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type = 10**

```
class msl.equipment.hislip.AsyncRemoteLocalResponse(control_code=0, parameter=0,
                                                    payload=b")
```

Bases: *Message*

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type = 11**

```
class msl.equipment.hislip.AsyncDeviceClear(control_code=0, parameter=0, payload=b")
```

Bases: *Message*

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type = 19**

```
class msl.equipment.hislip.AsyncDeviceClearAcknowledge(control_code=0, parameter=0,
                                                         payload=b'')
```

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type** = 23

#### property feature\_bitmap

The feature bitmap that the server prefers.

#### Type

*int*

```
class msl.equipment.hislip.DeviceClearComplete(control_code=0, parameter=0,
                                                  payload=b'')
```

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type** = 8

```
class msl.equipment.hislip.DeviceClearAcknowledge(control_code=0, parameter=0,
                                                    payload=b'')
```

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type** = 9

```
class msl.equipment.hislip.Trigger(control_code=0, parameter=0, payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 12

```
class msl.equipment.hislip.AsyncMaximumMessageSize(control_code=0, parameter=0,  
                                                    payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 15

```
class msl.equipment.hislip.AsyncMaximumMessageSizeResponse(control_code=0,  
                                                            parameter=0,  
                                                            payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 16

**property** **maximum\_message\_size**

The maximum message size that the server's synchronous channel accepts.

#### Type

[int](#)



```
class msl.equipment.hislip.GetDescriptors(control_code=0, parameter=0, payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 26

```
class msl.equipment.hislip.GetDescriptorsResponse(control_code=0, parameter=0,  
                                                payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 27

```
class msl.equipment.hislip.AsyncInitialize(control_code=0, parameter=0, payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 17

```
class msl.equipment.hislip.AsyncInitializeResponse(control_code=0, parameter=0,  
                                                  payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.

- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 18

**SECURE\_CONNECTION\_SUPPORTED** = 1

**property secure\_connection\_supported**

Whether secure connection capability is supported.

**Type**

`bool`

**property server\_vendor\_id**

The two-character vendor abbreviation of the server.

**Type**

`bytes`

**class** `msl.equipment.hislip.AsyncStatusQuery(control_code=0, parameter=0, payload=b'')`

Bases: `Message`

Create a new HiSLIP message.

#### Parameters

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 21

**class** `msl.equipment.hislip.AsyncStatusResponse(control_code=0, parameter=0, payload=b'')`

Bases: `Message`

Create a new HiSLIP message.

#### Parameters

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 22

**property status**

The status value.

**Type**

`int`

```
class msl.equipment.hislip.StartTLS(control_code=0, parameter=0, payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 28

```
class msl.equipment.hislip.AsyncStartTLS(control_code=0, parameter=0, payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 29

```
class msl.equipment.hislip.AsyncStartTLSResponse(control_code=0, parameter=0,
                                                  payload=b'')
```

Bases: [Message](#)

Create a new HiSLIP message.

#### Parameters

- **control\_code** ([int](#), optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** ([int](#), optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** ([bytes](#), optional) – The payload data.

**type** = 30

#### property busy

Whether the server is busy.

#### Type

[bool](#)

#### property success

Whether the request was successful.

**Type**`bool`**property error**

Whether there was an error processing the request.

**Type**`bool`

```
class msl.equipment.hislip.EndTLS(control_code=0, parameter=0, payload=b'')
```

Bases: [\*Message\*](#)

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 31

```
class msl.equipment.hislip.AsyncEndTLS(control_code=0, parameter=0, payload=b'')
```

Bases: [\*Message\*](#)

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 32

```
class msl.equipment.hislip.AsyncEndTLSResponse(control_code=0, parameter=0,  
                                              payload=b'')
```

Bases: [\*Message\*](#)

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type** = 33

**property busy**

Whether the server is busy.

**Type**

`bool`

**property success**

Whether the request was successful.

**Type**

`bool`

**property error**

Whether there was an error processing the request.

**Type**

`bool`

```
class msl.equipment.hislip.GetSaslMechanismList(control_code=0, parameter=0,
                                                payload=b'')
```

Bases: [\*Message\*](#)

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type = 34**

```
class msl.equipment.hislip.GetSaslMechanismListResponse(control_code=0,
                                                         parameter=0, payload=b'')
```

Bases: [\*Message\*](#)

Create a new HiSLIP message.

**Parameters**

- **control\_code** (`int`, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (`int`, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (`bytes`, optional) – The payload data.

**type = 35**

**property data**

List of SASL mechanisms.

**Type**

`list`

```
class msl.equipment.hislip.AuthenticationStart(control_code=0, parameter=0,  
                                              payload=b'')
```

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type** = 36

```
class msl.equipment.hislip.AuthenticationExchange(control_code=0, parameter=0,  
                                                  payload=b'')
```

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type** = 37

```
class msl.equipment.hislip.AuthenticationResult(control_code=0, parameter=0,  
                                                payload=b'')
```

Bases: *Message*

Create a new HiSLIP message.

#### Parameters

- **control\_code** (*int*, optional) – This 8-bit field is a general parameter for the message. If the field is not defined for a message, 0 shall be sent.
- **parameter** (*int*, optional) – This 32-bit field has various uses in different messages. If this field is not defined for a message, 0 shall be sent.
- **payload** (*bytes*, optional) – The payload data.

**type** = 38

#### property data

Additional data returned by the server.

#### Type

*bytes*

**property error**

Whether there was an error processing the request.

**Type**

`bool`

**property error\_code**

If authentication fails, the mechanism-dependent error code.

**Type**

`int`

**property success**

Whether the request was successful.

**Type**

`bool`

**class** `msl.equipment.hislip.HiSLIPClient(host)`

Bases: `object`

Base class for a HiSLIP client.

**Parameters**

**host** (`str`) – The hostname or IP address of the remote device.

**close()**

Close the TCP socket, if one is open.

**connect**(*port=4880, timeout=10*)

Connect to a specific port of the device.

**Parameters**

- **port** (`int`) – The port number to connect to.
- **timeout** (`float` or `None`, optional) – The maximum number of seconds to wait for the connection to be established.

**get\_descriptors()**

Descriptors were added in HiSLIP version 2.0 to provide extra information about specific server capabilities.

**Returns**

*GetDescriptorsResponse* – The response.

**property maximum\_server\_message\_size**

The maximum message size that the server accepts.

**Type**

`int`

**read**(*message, chunk\_size=4096*)

Read a message from the server.

**Parameters**

- **message** (*Message*) – An instance of the type of message to read.

- **chunk\_size** (*int*, optional) – The maximum number of bytes to receive at a time.

**Returns**

*Message* – The *message* that was passed in, but with its attributes updated with the information from the received data.

**get\_timeout()**

Get the socket timeout value.

**Returns**

*float* or *None* – The timeout, in seconds, of the socket.

**set\_timeout(timeout)**

Set the socket timeout value.

**Parameters**

**timeout** (*float* or *None*) – The timeout, in seconds, to use for the socket.

**property socket**

The reference to the socket.

**Type**

*socket*

**write(message)**

Write a message to the server.

**Parameters**

**message** (*Message*) – The message to write.

**class** msl.equipment.hislip.SyncClient(*host*)

Bases: *HiSLIPClient*

A synchronous connection to the HiSLIP server.

**Parameters**

**host** (*str*) – The hostname or IP address of the remote device.

**device\_clear\_complete(feature\_bitmap)**

Send the device-clear complete message.

Also resets the message id.

**Parameters**

**feature\_bitmap** (*int*) – The feature bitmap of the server (i.e., *AsyncDeviceClearAcknowledge.feature\_bitmap*).

**Returns**

*DeviceClearAcknowledge* – The response.

**initialize(major=1, minor=0, client\_id=b'XX', sub\_address=b'')**

Initialize the synchronous connection.

**Parameters**

- **major** (*int*, optional) – The major version number of the HiSLIP protocol that the client supports.



- **minor** (`int`, optional) – The minor version number of the HiSLIP protocol that the client supports.
- **client\_id** (`bytes`, optional) – The vendor ID of the client. Must have a length of 2 characters.
- **sub\_address** (`bytes`, optional) – A particular device managed by this server. For VISA clients this field corresponds to the VISA LAN device name (default is *hislip0*). The maximum length is 256 characters.

**Returns**

`InitializeResponse` – The response.

**property message\_id**

The id of the most-recent message that has completed.

**Type**

`int`

**property message\_id\_received**

The id of most-recent message that has been received from the server.

**Type**

`int`

**receive**(*size=None, max\_size=None, chunk\_size=4096*)

Receive data.

**Parameters**

- **size** (`int`, optional) – The number of bytes to read. If not specified, then read until a Response Message Terminator (RMT) is detected.
- **max\_size** (`int`, optional) – The maximum number of bytes that can be read. If not specified, then there is no limit.
- **chunk\_size** (`int`, optional) – The maximum number of bytes to receive at a time.

**Returns**

`bytearray` – The received data.

**property rmt**

`int` The current state of the Response Message Terminator (RMT).

**send**(*data*)

Send data with the Response Message Terminator (RMT) character.

**Parameters**

**data** (`bytes`) – The data to send.

**Returns**

`int` – The number of bytes sent.

**trigger**()

Send the trigger message (emulates a GPIB Group Execute Trigger event).

**start\_tls**()

Send the *StartTLS* message.

**end\_tls()**

Send the *EndTLS* message.

**get\_sasl\_mechanism\_list()**

Request the list of SASL mechanisms from the server.

**Returns**

*GetSaslMechanismListResponse* – The response.

**authentication\_start(*mechanism*)**

Send a SASL authentication method to the server.

**Parameters**

**mechanism** (*bytes*) – The selected mechanism to use for authentication.

**write\_authentication\_exchange(*data*)**

Send exchange data during the authentication transaction.

**Parameters**

**data** (*bytes*) – The data to send.

**read\_authentication\_exchange()**

Receive exchange data during the authentication transaction.

**Returns**

*AuthenticationExchange* – The exchange.

**authentication\_result()**

Receive an authentication result from the server.

**Returns**

*AuthenticationResult* – The result.

**class** msl.equipment.hislip.**AsyncClient**(*host*)

Bases: *HiSLIPClient*

An asynchronous connection to the HiSLIP server.

**Parameters**

**host** (*str*) – The hostname or IP address of the remote device.

**async\_initialize(*session\_id*)**

Initialize the asynchronous connection.

**Parameters**

**session\_id** (*int*) – The session ID.

**Returns**

*AsyncInitializeResponse* – The response.

**async\_maximum\_message\_size(*size*)**

Exchange the maximum message sizes that are accepted between the client and server.

**Parameters**

**size** (*int*) – The maximum message size that the client accepts.

**Returns**

*AsyncMaximumMessageSizeResponse* – The maximum message size that the server accepts.

**async\_lock\_request**(*timeout=None, lock\_string=""*)

Request a lock.

**Parameters**

- **timeout** (*float*, optional) – The number of seconds to wait to acquire a lock. A timeout of 0 indicates that the HiSLIP server should only grant the lock if it is available immediately.
- **lock\_string** (*str*, optional) – An ASCII string that identifies this lock. If not specified, then an exclusive lock is requested, otherwise the string indicates an identification of a shared-lock request. The maximum length is 256 characters.

**Returns**

*AsyncLockResponse* – The response.

**async\_lock\_release**(*message\_id*)

Release a lock.

**Parameters**

**message\_id** (*int*) – The most recent message id that was completed on the synchronous channel (i.e., *SyncClient.message\_id*).

**Returns**

*AsyncLockResponse* – The response.

**async\_lock\_info**()

Request the lock status from the HiSLIP server.

**Returns**

*AsyncLockInfoResponse* – The response.

**async\_remote\_local\_control**(*request, message\_id*)

Send a GPIB-like remote/local control request.

**Parameters**

- **request** (*int*) – The request to perform.
  - 0 – Disable remote, *VI\_GPIB\_REN\_DEASSERT*
  - 1 – Enable remote, *VI\_GPIB\_REN\_ASSERT*
  - 2 – Disable remote and go to local, *VI\_GPIB\_REN\_DEASSERT\_GTL*
  - 3 – Enable Remote and go to remote, *VI\_GPIB\_REN\_ASSERT\_ADDRESS*
  - 4 – Enable remote and lock out local, *VI\_GPIB\_REN\_ASSERT\_LLO*
  - 5 – Enable remote, go to remote, and set local lockout, *VI\_GPIB\_REN\_ASSERT\_ADDRESS\_LLO*
  - 6 – go to local without changing REN or lockout state, *VI\_GPIB\_REN\_ADDRESS\_GTL*
- **message\_id** (*int*) – The most recent message id that was completed on the synchronous channel (i.e., *SyncClient.message\_id*).

**Returns**

*AsyncRemoteLocalResponse* – The response.

**async\_device\_clear()**

Send the device clear request.

**Returns**

*AsyncDeviceClearAcknowledge* – The response.

**async\_status\_query(synchronous)**

Status query transaction.

The status query provides an 8-bit status response from the server that corresponds to the VISA *viReadSTB* operation.

**Parameters**

**synchronous** (*SyncClient*) – The synchronous client that corresponds with this asynchronous client.

**Returns**

*AsyncStatusResponse* – The response.

**async\_start\_tls(synchronous)**

Initiate the secure connection transaction.

**Parameters**

**synchronous** (*SyncClient*) – The synchronous client that corresponds with this asynchronous client.

**Returns**

*AsyncStartTLSResponse* – The response.

**async\_end\_tls(synchronous)**

Initiate the end of the secure connection transaction.

**Parameters**

**synchronous** (*SyncClient*) – The synchronous client that corresponds with this asynchronous client.

**Returns**

*AsyncEndTLSResponse* – The response.

## msl.equipment.record\_types module

Records from *Equipment-Register Database*'s or *Connections Database*'s.

```
class msl.equipment.record_types.EquipmentRecord(alias="", calibrations=None,
                                                  category="", connection=None,
                                                  description="", is_operable=False,
                                                  maintenances=None, manufacturer="",
                                                  model="", serial="", team="",
                                                  unique_key="", **user_defined)
```

Bases: *Record*

Contains the information about an equipment record in an *Equipment-Register Database*.

**Parameters**

- **alias** (*str*) – An alias to use to reference this equipment by.
- **calibrations** (*list* of *CalibrationRecord*) – The calibration history of the equipment.
- **category** (*str*) – The category (e.g., Laser, DMM) that the equipment belongs to.
- **connection** (*ConnectionRecord*) – The information necessary to communicate with the equipment.
- **description** (*str*) – A description about the equipment.
- **is\_operable** (*bool*) – Whether the equipment is able to be used.
- **maintenances** (*list* of *MaintenanceRecord*) – The maintenance history of the equipment.
- **manufacturer** (*str*) – The name of the manufacturer of the equipment.
- **model** (*str*) – The model number of the equipment.
- **serial** (*str*) – The serial number (or unique identifier) of the equipment.
- **team** (*str*) – The team (e.g., Light Standards) that the equipment belongs to.
- **unique\_key** (*str*) – The key that uniquely identifies the equipment record in a database.
- **\*\*user\_defined** – All additional key-value pairs are added to the *user\_defined* attribute.

#### **alias**

An alias to use to reference this equipment by.

The *alias* can be defined in 4 ways:

- by specifying it when the EquipmentRecord is created
- by setting the value after the EquipmentRecord has been created
- in the **<equipment>** XML tag in a *Configuration File*
- in the **Properties** field in a *Connections Database*

#### **Type**

*str*

#### **calibrations**

The calibration history of the equipment.

#### **Type**

*tuple* of *CalibrationRecord*

#### **category**

The category (e.g., Laser, DMM) that the equipment belongs to.

#### **Type**

*str*

**description**

A description about the equipment.

**Type**

`str`

**is\_operable**

Whether the equipment is able to be used.

**Type**

`bool`

**maintenances**

The maintenance history of the equipment.

**Type**

`tuple` of *MaintenanceRecord*

**manufacturer**

The name of the manufacturer of the equipment.

**Type**

`str`

**model**

The model number of the equipment.

**Type**

`str`

**serial**

The serial number (or unique identifier) of the equipment.

**Type**

`str`

**connection**

The information necessary to communicate with the equipment.

**Type**

*ConnectionRecord*

**team**

The team (e.g., Light Standards) that the equipment belongs to.

**Type**

`str`

**unique\_key**

The key that uniquely identifies the equipment record in a database.

**Type**

`str`

**user\_defined**

User-defined, key-value pairs.

**Type**

*RecordDict*

**connect**(*demo=None*)

Establish a connection to the equipment.

Calls the `connect()` function.

**Parameters**

**demo** (`bool`, optional) – Whether to simulate a connection to the equipment by opening a connection in demo mode. This allows you to test your code if the equipment is not physically connected to a computer.

If `None` then the *demo* value is determined from the `DEMO_MODE` attribute.

**Returns**

A `Connection` subclass.

**is\_calibration\_due**(*months=0*)

Whether the equipment needs to be re-calibrated.

**Parameters**

**months** (`int`, optional) – The number of months to add to today’s date to determine if the equipment needs to be re-calibrated within a certain amount of time. For example, if `months = 6` then that is a way of asking “*is a re-calibration due within the next 6 months?*”.

**Returns**

`bool` – `True` if the equipment needs to be re-calibrated, `False` if it does not need to be re-calibrated (or it has never been calibrated).

**property latest\_calibration**

The latest calibration or `None` if the equipment has never been calibrated.

**Type**

`CalibrationRecord`

**next\_calibration\_date**()

The date that the next calibration is due.

**Returns**

`datetime.date` – The next calibration date (or `None` if the equipment has never been calibrated or if it is no longer in operation).

**to\_dict**()

Convert this `EquipmentRecord` to a `dict`.

**Returns**

`dict` – The `EquipmentRecord` as a `dict`.

**to\_json**()

Convert this `EquipmentRecord` to be JSON serializable.

**Returns**

`dict` – The `EquipmentRecord` as a JSON-serializable object.

**to\_xml**()

Convert this `EquipmentRecord` to an XML `Element`.

**Returns**

`Element` – The `EquipmentRecord` as an XML element.

```
class msl.equipment.record_types.CalibrationRecord(calibration_cycle=0,
                                                    calibration_date=None,
                                                    measurands=None,
                                                    report_date=None,
                                                    report_number="")
```

Bases: [Record](#)

Contains the information about a calibration record in an [Equipment-Register Database](#).

#### Parameters

- **calibration\_cycle** ([int](#) or [float](#)) – The number of years that can pass before the equipment must be re-calibrated.
- **calibration\_date** ([datetime.date](#), [datetime.datetime](#) or [str](#)) – The date that the calibration was performed. If a [str](#) then in the format 'YYYY-MM-DD'.
- **measurands** ([list](#) of [MeasurandRecord](#)) – The quantities that were measured.
- **report\_date** ([datetime.date](#), [datetime.datetime](#) or [str](#)) – The date that the report was issued. If a [str](#) then in the format 'YYYY-MM-DD'.
- **report\_number** ([str](#)) – The report number.

#### to\_dict()

[dict](#): Convert the Record to a [dict](#).

#### calibration\_cycle

The number of years that can pass before the equipment must be re-calibrated.

##### Type

[float](#)

#### calibration\_date

The date that the calibration was performed.

##### Type

[datetime.date](#)

#### measurands

The quantities that were measured.

##### Type

[RecordDict](#)

#### report\_date

The date that the report was issued.

##### Type

[datetime.date](#)

#### report\_number

The report number.

##### Type

[str](#)



**to\_json()**

Convert this *CalibrationRecord* to be JSON serializable.

**Returns**

*dict* – The *CalibrationRecord* as a JSON-serializable object.

**to\_xml()**

Convert this *CalibrationRecord* to an XML *Element*.

**Returns**

*Element* – The *CalibrationRecord* as a XML *Element*.

```
class msl.equipment.record_types.MeasurandRecord(calibration=None, conditions=None,
                                                  type="", unit="")
```

Bases: *Record*

Contains the information about a measurement for a calibration.

**Parameters**

- **calibration** (*dict*) – The information about the calibration.
- **conditions** (*dict*) – The information about the conditions under which the measurement was performed.
- **type** (*str*) – The type of measurement (e.g., voltage, temperature, transmittance, ...).
- **unit** (*str*) – The unit that is associated with the measurement (e.g., V, deg C, %, ...).

**to\_dict()**

*dict*: Convert the Record to a *dict*.

**calibration**

The information about calibration.

**Type**

*RecordDict*

**conditions**

The information about the measurement conditions.

**Type**

*RecordDict*

**type**

The type of measurement (e.g., voltage, temperature, transmittance, ...).

**Type**

*str*

**unit**

The unit that is associated with the measurement (e.g., V, deg C, %, ...).

**Type**

*str*

**to\_json()**

Convert this *MeasurandRecord* to be JSON serializable.

**Returns**

*dict* – The *MeasurandRecord* as a JSON-serializable object.

**to\_xml()**

Convert this *MeasurandRecord* to an XML *Element*.

**Returns**

*Element* – The *MeasurandRecord* as a XML *Element*.

**class** msl.equipment.record\_types.**MaintenanceRecord**(comment="", date=None)

Bases: *Record*

Contains the information about a maintenance record in an *Equipment-Register Database*.

**Parameters**

- **comment** (*str*) – A description of the maintenance that was performed.
- **date** (*datetime.date*, *datetime.datetime* or *str*) – An object that can be converted to a *datetime.date* object. If a *str* then in the format 'YYYY-MM-DD'.

**comment**

A description of the maintenance that was performed.

**Type**

*str*

**date**

The date that the maintenance was performed.

**Type**

*datetime.date*

**to\_json()**

Convert this *MaintenanceRecord* to be JSON serializable.

**Returns**

*dict* – The *MaintenanceRecord* as a JSON-serializable object.

**to\_xml()**

Convert this *MaintenanceRecord* to an XML *Element*.

**Returns**

*Element* – The *MaintenanceRecord* as a XML *Element*.

**to\_dict()**

*dict*: Convert the Record to a *dict*.

**class** msl.equipment.record\_types.**ConnectionRecord**(address="", backend=*Backend.MSL*,  
interface=None, manufacturer="",  
model="", serial="", \*\*properties)

Bases: *Record*

Contains the information about a connection record in a *Connections Database*.

### Parameters

- **address** (*str*) – The address to use for the connection (see *Address Syntax* for examples).
- **backend** (*str*, *int*, or *Backend*) – The backend to use to communicate with the equipment. The value must be able to be converted to a *Backend* enum.
- **interface** (*str*, *int*, or *Interface*) – The interface to use to communicate with the equipment. If *None* then determines the *interface* based on the value of *address*. If specified then the value must be able to be converted to a *Interface* enum.
- **manufacturer** (*str*) – The name of the manufacturer of the equipment.
- **model** (*str*) – The model number of the equipment.
- **serial** (*str*) – The serial number (or unique identifier) of the equipment.
- **properties** – Additional key-value pairs that are required to communicate with the equipment.

#### address

The address to use for the connection (see *Address Syntax* for examples).

##### Type

*str*

#### backend

The backend to use to communicate with the equipment.

##### Type

*Backend*

#### interface

The interface that is used for the communication system that transfers data between a computer and the equipment (only used if the *backend* is equal to *MSL*).

##### Type

*Interface*

#### manufacturer

The name of the manufacturer of the equipment.

##### Type

*str*

#### model

The model number of the equipment.

##### Type

*str*

#### properties

Additional key-value pairs that are required to communicate with the equipment.

For example, communicating via RS-232 may require:

```
{'baud_rate': 19200, 'parity': 'even'}
```

See the *Connections Database* for examples on how to set the *properties*.

**Type**

`dict`

**serial**

The serial number (or unique identifier) of the equipment.

**Type**

`str`

**to\_json()**

Convert this *ConnectionRecord* to be JSON serializable.

**Returns**

`dict` – The *ConnectionRecord* as a JSON-serializable object.

**to\_xml()**

Convert this *ConnectionRecord* to an XML Element.

**Returns**

`Element` – The *ConnectionRecord* as a XML Element.

**to\_dict()**

`dict`: Convert the Record to a `dict`.

**class** `msl.equipment.record_types.RecordDict(dictionary)`

Bases: `Mapping`

A read-only dictionary that supports attribute access via a key lookup.

**copy()**

*RecordDict*: Return a copy of the *RecordDict*.

**to\_xml(tag='RecordDict')**

Convert the *RecordDict* to an XML Element

**Parameters**

**tag** (`str`) – The name of the Element.

**Returns**

`Element` – The *RecordDict* as an XML Element.

**to\_json()**

`dict`: Convert the *RecordDict* to be JSON serializable.

**class** `msl.equipment.record_types.Record`

Bases: `object`

**to\_dict()**

`dict`: Convert the Record to a `dict`.

**to\_json()**

`dict`: Convert the Record to be JSON serializable.

This differs from `to_dict()` such that all values that are not JSON serializable, like `datetime.date` objects, are converted to a `str`.

`to_xml()`

**Element:** Convert the Record to an XML **Element**.

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

MSL resources for connecting to equipment.

`msl.equipment.resources.register(manufacturer=None, model=None, flags=0)`

Use as a decorator to register a resource class.

### **Parameters**

- **manufacturer** (**str**, optional) – The name of the manufacturer. Can be a regex pattern.
- **model** (**str**, optional) – The model number of the equipment. Can be a regex pattern.
- **flags** (**int**, optional) – The flags to use for the regex pattern.

`msl.equipment.resources.find_resource_class(record)`

Find the resource class for this *record*.

### **Parameters**

**record** (*EquipmentRecord* or *ConnectionRecord*) – A record type. If the *properties* attribute contains a *resource\_class\_name* key with a value that is equal to the name of a resource class it forces this resource class to be returned, provided that a resource class with the requested name exists.

### **Returns**

The *Connection* subclass or *None* if a resource class cannot be found.

## **Submodules**

### **msl.equipment.resources.dmm module**

Encapsulate some of the commands to communicate with a Digital Multimeter.

`msl.equipment.resources.dmm.dmm_factory(connection_record, connection_class)`

Returns a class that encapsulates some of the commands to communicate with a Digital Multimeter.

*To add a DMM to the list of supported DMM's see the source code of this module to follow the template.*

This function is not meant to be called directly. Use the *connect()* method to connect to the equipment.

### **Parameters**

- **connection\_record** (*ConnectionRecord*) – A connection record from a *Connections Database*.
- **connection\_class** (*ConnectionMessageBased* or *MessageBasedResource*) – A connection subclass that communicates with the equipment through *read* and *write* commands.

**Returns**

*Connection* – The *connection\_class* that was passed in with additional methods for communicating with the DMM, provided that the model number of the DMM is one of the DMM's that is supported. Otherwise returns the original, unmodified *connection\_class* object.

**msl.equipment.resources.utils module**

Utility functions/classes to help create modules in the **msl.equipment.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**(*path*, *remove\_comments=True*)

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 comments.

**Parameters**

- **path** (*str*) – The path to a C/C++ header file.
- **remove\_comments** (*bool*, optional) – Whether to remove the comments.

**Returns**

*list* of *str* – The list of lines in the header file.

**class msl.equipment.resources.utils.CHeader**(*path*, *remove\_comments=True*)

Bases: *object*

Parses a C/C++ header file to determine the constants, enums, structs, callbacks and the function signatures.

**Parameters**

- **path** (*str*) – The path to the header file.
- **remove\_comments** (*bool*, optional) – Whether to remove the comments.

**constants**(*ignore\_ifdef=True*)

Finds the *#define* statements that are in the C/C++ header file.

**Parameters**

**ignore\_ifdef** (*bool*, optional) – Whether to ignore the *#define* statements in between the *#ifdef* and *#endif* statements.

**Returns**

*dict* – 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 enum name, the enum data type, 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 contain 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 (ps60000openUnit)(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, see `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*.

## Subpackages

### `msl.equipment.resources.aim_tti` package

Resources for equipment from [Aim and Thurlby Thandar Instruments](#).

## Submodules

### `msl.equipment.resources.aim_tti.mx_series` module

Establishes a connection to an MX100QP, MX100TP or MX180TP DC power supply from [Aim and Thurlby Thandar Instruments](#)

**class** `msl.equipment.resources.aim_tti.mx_series.MXSeries(record)`

Bases: `object`

Establishes a connection to an MX100QP, MX100TP or MX180TP DC power supply from [Aim and Thurlby Thandar Instruments](#) for different interfaces:

- `Interface.PROLOGIX`
- `Interface.SERIAL`
- `Interface.SOCKET`

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**clear()**

Send the clear, \*CLS, command.

**decrement\_current(channel)**

Decrement the current limit by step size of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**See also:**

`set_current_step_size()`



**decrement\_voltage**(*channel*, *verify=True*)

Decrement the voltage by step size of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **verify** (*bool*, optional) – Whether to verify that the output voltage has stabilized at the decremented voltage before returning to the calling program.

See also:

[`set\_voltage\_step\_size\(\)`](#)

**event\_status\_register**(*as\_integer=True*)

Read and clear the standard event status register.

**Parameters**

**as\_integer** (*bool*, optional) – Whether to return the value as an *int*.

**Returns**

*int* or *str* – The event status register value. The data type depends on the value of *as\_integer*. If a *str* is returned then it will have a length of 8. For example,

- '10000000' or the integer value 128
- '00100000' or the integer value 32

**get\_current**(*channel*)

Get the output current of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* – The output current, in Amps.

**get\_current\_limit**(*channel*)

Get the current limit of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* – The current limit, in Amps.

**get\_current\_step\_size**(*channel*)

Get the current limit step size of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* – The current limit step size, in Amps.

**get\_over\_current\_protection**(*channel*)

Get the over-current protection trip point of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* or *None* – If the trip point is enabled then returns the trip point value, in Amps. Returns *None* if the over-current protection is disabled.

**get\_over\_voltage\_protection(channel)**

Get the over-voltage protection trip point of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* or *None* – If the trip point is enabled then returns the trip point value, in Volts. Returns *None* if the over-voltage protection is disabled.

**get\_voltage(channel)**

Get the output voltage of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* – The output voltage, in Volts.

**get\_voltage\_range(channel)**

Get the output voltage range index of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*int* – The output voltage range index. See the manual for more details. For example, 2 = 35V/3A.

**get\_voltage\_setpoint(channel)**

Get the set-point voltage of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* – The set-point voltage, in Volts.

**get\_voltage\_step\_size(channel)**

Get the voltage step size of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*float* – The voltage step size, in Volts.

**get\_voltage\_tracking\_mode()**

Get the voltage tracking mode of the unit.

**Returns**

*int* – The voltage tracking mode. See the manual for more details.

**increment\_current**(*channel*)

Increment the current limit by step size of the output channel.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

See also:

[`set\_current\_step\_size\(\)`](#)

**increment\_voltage**(*channel*, *verify=True*)

Increment the voltage by step size of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **verify** (*bool*, optional) – Whether to verify that the output voltage has stabilized at the incremented voltage before returning to the calling program.

See also:

[`set\_voltage\_step\_size\(\)`](#)

**is\_output\_on**(*channel*)

Check if the output channel is on or off.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**Returns**

*bool* – Whether the output channel is on (*True*) or off (*False*).

**turn\_on**(*channel*)

Turn the output channel on.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**turn\_on\_multi**(*options=None*)

Turn multiple output channels on (the Multi-On feature).

**Parameters**

**options** (*dict*, optional) – Set the Multi-On option for each output channel before setting Multi-On. If not specified then uses the pre-programmed options. If a particular output channel is not included in *options* then uses the pre-programmed option for that channel. The keys are the output channel number and the value can be *False* (set the channel to NEVER, see the manual for more details), *True* (set the channel to QUICK, see the manual for more details) or a delay in milliseconds (as an *int*).

Examples:

- {1: *False*} → channel 1 does not turn on
- {2: 100} → channel 2 has a 100-ms delay
- {1: 100, 3: *True*} → channel 1 has a 100-ms delay and channel 3 turns on immediately

- {1: 100, 2: 200, 3: 300} → channel 1 has a 100-ms delay, channel 2 has a 200-ms delay and channel 3 has a 300-ms delay

See also:

`set_multi_on_delay()`, `set_multi_on_action()`

**turn\_off**(*channel*)

Turn the output channel off.

**Parameters**

**channel** (*int*) – The output channel. The first output channel is 1 (not 0).

**turn\_off\_multi**(*options=None*)

Turn multiple output channels off (the Multi-Off feature).

**Parameters**

**options** (*dict*, optional) – Set the Multi-Off option for each output channel before setting Multi-Off. If not specified then uses the pre-programmed options. If a particular output channel is not included in *options* then uses the pre-programmed option for that channel. The keys are the output channel number and the value can be **False** (set the channel to NEVER, see the manual for more details), **True** (set the channel to QUICK, see the manual for more details) or a delay in milliseconds (as an *int*).

Examples:

- {1: False} → channel 1 does not turn off
- {2: 100} → channel 2 has a 100-ms delay
- {1: 100, 3: True} → channel 1 has a 100-ms delay and channel 3 turns off immediately
- {1: 100, 2: 200, 3: 300} → channel 1 has a 100-ms delay, channel 2 has a 200-ms delay and channel 3 has a 300-ms delay

See also:

`set_multi_off_delay()`, `set_multi_off_action()`

**recall**(*channel, index*)

Recall the settings of the output channel from the store.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **index** (*int*) – The store index number, can be 0-49.

See also:

`save()`

**recall\_all**(*index*)

Recall the settings for all output channels from the store.

**Parameters**

**index** (*int*) – The store index number, can be 0-49.

See also:

[`save\_all\(\)`](#)

**reset()**

Send the reset, \*RST, command.

**reset\_trip()**

Attempt to clear all trip conditions.

**save(channel, index)**

Save the present settings of the output channel to the store.

**Parameters**

- **channel** (`int`) – The output channel. The first output channel is 1 (not 0).
- **index** (`int`) – The store index number, can be 0-49.

See also:

[`recall\(\)`](#)

**save\_all(index)**

Save the settings of all output channels to the store.

**Parameters**

- **index** (`int`) – The store index number, can be 0-49.

See also:

[`recall\_all\(\)`](#)

**set\_current\_limit(channel, value)**

Set the current limit of the output channel.

**Parameters**

- **channel** (`int`) – The output channel. The first output channel is 1 (not 0).
- **value** (`float`) – The current limit, in Amps.

**set\_current\_meter\_averaging(channel, value)**

Set the current meter measurement averaging of the output channel.

**Parameters**

- **channel** (`int`) – The output channel. The first output channel is 1 (not 0).
- **value** (`str`) – Can be ON, OFF, LOW, MED or HIGH.

**set\_current\_step\_size(channel, size)**

Set the current limit step size of the output channel.

**Parameters**

- **channel** (`int`) – The output channel. The first output channel is 1 (not 0).
- **size** (`float`) – The current limit step size, in Amps.

**set\_multi\_on\_action**(*channel, action*)

Set the Multi-On action of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **action** (*str*) – The Multi-On action, one of QUICK, NEVER or DELAY.

**set\_multi\_on\_delay**(*channel, delay*)

Set the Multi-On delay, in milliseconds, of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **delay** (*int*) – The delay, in milliseconds.

**set\_multi\_off\_action**(*channel, action*)

Set the Multi-Off action of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **action** (*str*) – The Multi-Off action, one of QUICK, NEVER or DELAY.

**set\_multi\_off\_delay**(*channel, delay*)

Set the Multi-Off delay, in milliseconds, of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **delay** (*int*) – The delay, in milliseconds.

**set\_over\_current\_protection**(*channel, enable, value=None*)

Set the over-current protection trip point of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **enable** (*bool*) – Whether to enable (*True*) or disable (*False*) the over-current protection trip point.
- **value** (*float*, optional) – If the trip point is enabled then you must specify a value, in Amps.

**set\_over\_voltage\_protection**(*channel, enable, value=None*)

Set the over-voltage protection trip point of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **enable** (*bool*) – Whether to enable (*True*) or disable (*False*) the over-voltage protection trip point.
- **value** (*float*, optional) – If the trip point is enabled then you must specify a value, in Volts.

**set\_voltage**(*channel*, *value*, *verify=True*)

Set the output voltage of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **value** (*float*) – The value, in Volts.
- **verify** (*bool*, optional) – Whether to verify that the output voltage has stabilized at *value* before returning to the calling program.

**set\_voltage\_range**(*channel*, *index*)

Set the output voltage range of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **index** (*int*) – The output voltage range index. See the manual for more details. For example, 2 = 35V/3A.

**set\_voltage\_step\_size**(*channel*, *size*)

Set the voltage step size of the output channel.

**Parameters**

- **channel** (*int*) – The output channel. The first output channel is 1 (not 0).
- **size** (*float*) – The voltage step size, in Volts.

**set\_voltage\_tracking\_mode**(*mode*)

Set the voltage tracking mode of the unit.

**Parameters**

- **mode** (*int*) – The voltage tracking mode. See the manual for more details.

## **msl.equipment.resources.avantes package**

Wrapper around the `avaspec.dll` SDK from Avantes.

### **Submodules**

#### **msl.equipment.resources.avantes.avaspec module**

Wrapper around the `avaspec.dll` SDK from Avantes.

The wrapper was written using v9.7.0.0 of the SDK.

```
class msl.equipment.resources.avantes.avaspec.DeviceStatus(value, names=None, *,  
                                                         module=None,  
                                                         qualname=None,  
                                                         type=None, start=1,  
                                                         boundary=None)
```

Bases: `IntEnum`

DeviceStatus enum.

UNKNOWN = 0

USB\_AVAILABLE = 1

USB\_IN\_USE\_BY\_APPLICATION = 2

USB\_IN\_USE\_BY\_OTHER = 3

ETH\_AVAILABLE = 4

ETH\_IN\_USE\_BY\_APPLICATION = 5

ETH\_IN\_USE\_BY\_OTHER = 6

ETH\_ALREADY\_IN\_USE\_USB = 7

```
class msl.equipment.resources.avantes.avaspec.InterfaceType(value, names=None, *,
                                                            module=None,
                                                            qualname=None,
                                                            type=None, start=1,
                                                            boundary=None)
```

Bases: `IntEnum`

InterfaceType enum.

RS232 = 0

USB5216 = 1

USBMINI = 2

USB7010 = 3

ETH7010 = 4

```
class msl.equipment.resources.avantes.avaspec.SensType(value, names=None, *,
                                                         module=None,
                                                         qualname=None, type=None,
                                                         start=1, boundary=None)
```

Bases: `IntEnum`

SensType enum.

SENS\_HAMS8378\_256 = 1

SENS\_HAMS8378\_1024 = 2

SENS\_ILX554 = 3

SENS\_HAMS9201 = 4

SENS\_TCD1304 = 5

SENS\_TSL1301 = 6

SENS\_TSL1401 = 7

SENS\_HAMS8378\_512 = 8



**SENS\_HAMS9840** = 9  
**SENS\_ILX511** = 10  
**SENS\_HAMS10420\_2048X64** = 11  
**SENS\_HAMS11071\_2048X64** = 12  
**SENS\_HAMS7031\_1024X122** = 13  
**SENS\_HAMS7031\_1024X58** = 14  
**SENS\_HAMS11071\_2048X16** = 15  
**SENS\_HAMS11155\_2048** = 16  
**SENS\_SU256LSB** = 17  
**SENS\_SU512LDB** = 18  
**SENS\_HAMS11638** = 21  
**SENS\_HAMS11639** = 22  
**SENS\_HAMS12443** = 23  
**SENS\_HAMG9208\_512** = 24  
**SENS\_HAMG13913** = 25  
**SENS\_HAMS13496** = 26

**class** msl.equipment.resources.avantes.avaspec.**AvsIdentityType**

Bases: Structure

IdentityType Structure.

**SerialNumber**

Structure/Union member

**Status**

Structure/Union member

**UserFriendlyName**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.**BroadcastAnswerType**

Bases: Structure

BroadcastAnswerType Structure.

**InterfaceType**

Structure/Union member

**LocalIp**

Structure/Union member

**RemoteHostIp**

Structure/Union member

**port**

Structure/Union member

**reserved**

Structure/Union member

**serial**

Structure/Union member

**status**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.**ControlSettingsType**

Bases: Structure

ControlSettingsType Structure.

**m\_LaserDelay**

Structure/Union member

**m\_LaserWaveLength**

Structure/Union member

**m\_LaserWidth**

Structure/Union member

**m\_StoreToRam**

Structure/Union member

**m\_StrobeControl**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.**DarkCorrectionType**

Bases: Structure

DarkCorrectionType Structure.

**m\_Enable**

Structure/Union member

**m\_ForgetPercentage**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.**DetectorType**

Bases: Structure

DetectorType Structure.

**m\_DefectivePixels**

Structure/Union member

**m\_ExtOffset**

Structure/Union member

**m\_Gain**

Structure/Union member

**m\_NLEnable**

Structure/Union member

**m\_NrPixels**

Structure/Union member

**m\_Offset**

Structure/Union member

**m\_Reserved**

Structure/Union member

**m\_SensorType**

Structure/Union member

**m\_aFit**

Structure/Union member

**m\_aHighNLCounts**

Structure/Union member

**m\_aLowNLCounts**

Structure/Union member

**m\_aNLCorrect**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.SmoothingType

Bases: Structure

SmoothingType Structure.

**m\_SmoothModel**

Structure/Union member

**m\_SmoothPix**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.SpectrumCalibrationType

Bases: Structure

SpectrumCalibrationType Structure.

**m\_CalInttime**

Structure/Union member

**m\_Smoothing**

Structure/Union member

**m\_aCalibConvers**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.IrradianceType

Bases: Structure

IrradianceType Structure.

**m\_CalibrationType**

Structure/Union member

**m\_FiberDiameter**

Structure/Union member

**m\_IntensityCalib**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.SpectrumCorrectionType

Bases: Structure

SpectrumCorrectionType Structure.

**m\_aSpectrumCorrect**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.TriggerType

Bases: Structure

TriggerType Structure.

**m\_Mode**

Structure/Union member

**m\_Source**

Structure/Union member

**m\_SourceType**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.MeasConfigType

Bases: Structure

MeasConfigType Structure.

**m\_Control**

Structure/Union member

**m\_CorDynDark**

Structure/Union member

**m\_IntegrationDelay**

Structure/Union member

**m\_IntegrationTime**

Structure/Union member

**m\_NrAverages**

Structure/Union member

**m\_SaturationDetection**

Structure/Union member

**m\_Smoothing**

Structure/Union member

**m\_StartPixel**

Structure/Union member

**m\_StopPixel**

Structure/Union member

**m\_Trigger**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.TimeStampType

Bases: Structure

TimeStampType Structure.

**m\_Date**

Structure/Union member

**m\_Time**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.StandAloneType

Bases: Structure

StandAloneType Structure.

**m\_Enable**

Structure/Union member

**m\_Meas**

Structure/Union member

**m\_Nmsr**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.DynamicStorageType

Bases: Structure

DynamicStorageType Structure.

**m\_Nmsr**

Structure/Union member

**m\_Reserved**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.TempSensorType

Bases: Structure

TempSensorType Structure.

**m\_aFit**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.TecControlType

Bases: Structure

TecControlType Structure.

**m\_Enable**

Structure/Union member

**m\_Setpoint**

Structure/Union member

**m\_aFit**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.ProcessControlType

Bases: Structure

ProcessControlType Structure.

**m\_AnalogHigh**

Structure/Union member

**m\_AnalogLow**

Structure/Union member

**m\_DigitalHigh**

Structure/Union member

**m\_DigitalLow**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.EthernetSettingsType

Bases: Structure

EthernetSettingsType Structure.

**m\_DhcpEnabled**

Structure/Union member

**m\_Gateway**

Structure/Union member

**m\_IpAddr**

Structure/Union member

**m\_LinkStatus**

Structure/Union member

**m\_NetMask**

Structure/Union member

**m\_TcpPort**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.OemDataType

Bases: Structure

OemDataType Structure.

**m\_data**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.HeartbeatRespType

Bases: Structure

HeartbeatRespType Structure.

**m\_BitMatrix**

Structure/Union member

**m\_Reserved**

Structure/Union member

**class** msl.equipment.resources.avantes.avaspec.DeviceConfigType

Bases: Structure

DeviceConfigType Structure.

**m\_ConfigVersion**

Structure/Union member

**m\_Detector**

Structure/Union member

**m\_DynamicStorage**

Structure/Union member

**m\_EthernetSettings**

Structure/Union member

**m\_Irradiance**

Structure/Union member

**m\_Len**

Structure/Union member

**m\_OemData**

Structure/Union member

**m\_ProcessControl**

Structure/Union member

**m\_Reflectance**

Structure/Union member

**m\_SpectrumCorrect**

Structure/Union member

**m\_StandAlone**

Structure/Union member

**m\_TecControl**

Structure/Union member

**m\_aReserved**

Structure/Union member

**m\_aTemperature**

Structure/Union member

**m\_aUserFriendlyId**

Structure/Union member

**msl.equipment.resources.avantes.avaspec.MeasureCallback**

Used as a decorator for a callback function when a scan is available.

**class** msl.equipment.resources.avantes.avaspec.**Avantes**(*record*)Bases: [ConnectionSDK](#)

Wrapper around the avaspec.dll SDK from Avantes.

The [properties](#) for an Avantes connection supports the following key-value pairs in the [Connections Database](#):

```
'port_id': int, One of -1 (Ethernet+USB), 0 (USB) or 256 (Ethernet),  
↪ [default: -1]  
'activate': bool, Whether to automatically activate the connection,  
↪ [default: True]
```

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.**Parameters****record** ([EquipmentRecord](#)) – A record from an [Equipment-Register Database](#).**WM\_MEAS\_READY** = 32769**SETTINGS\_RESERVED\_LEN** = 9720**INVALID\_AVS\_HANDLE\_VALUE** = 1000**USER\_ID\_LEN** = 64**AVS\_SERIAL\_LEN** = 10**MAX\_TEMP\_SENSORS** = 3**ROOT\_NAME\_LEN** = 6**VERSION\_LEN** = 16**AVASPEC\_ERROR\_MSG\_LEN** = 8**AVASPEC\_MIN\_MSG\_LEN** = 6**OEM\_DATA\_LEN** = 4096**NR\_WAVELEN\_POL\_COEF** = 5**NR\_NONLIN\_POL\_COEF** = 8**MAX\_VIDEO\_CHANNELS** = 2**NR\_DEFECTIVE\_PIXELS** = 30**MAX\_NR\_PIXELS** = 4096**NR\_TEMP\_POL\_COEF** = 5



```
NR_DAC_POL_COEF = 2
SAT_PEAK_INVERSION = 2
SW_TRIGGER_MODE = 0
HW_TRIGGER_MODE = 1
SS_TRIGGER_MODE = 2
EXTERNAL_TRIGGER = 0
SYNC_TRIGGER = 1
EDGE_TRIGGER_SOURCE = 0
LEVEL_TRIGGER_SOURCE = 1
ILX_FIRST_USED_DARK_PIXEL = 2
ILX_USED_DARK_PIXELS = 14
ILX_TOTAL_DARK_PIXELS = 18
TCD_FIRST_USED_DARK_PIXEL = 0
TCD_USED_DARK_PIXELS = 12
TCD_TOTAL_DARK_PIXELS = 13
HAMS9840_FIRST_USED_DARK_PIXEL = 0
HAMS9840_USED_DARK_PIXELS = 8
HAMS9840_TOTAL_DARK_PIXELS = 8
HAMS10420_FIRST_USED_DARK_PIXEL = 0
HAMS10420_USED_DARK_PIXELS = 4
HAMS10420_TOTAL_DARK_PIXELS = 4
HAMS11071_FIRST_USED_DARK_PIXEL = 0
HAMS11071_USED_DARK_PIXELS = 4
HAMS11071_TOTAL_DARK_PIXELS = 4
HAMS7031_FIRST_USED_DARK_PIXEL = 0
HAMS7031_USED_DARK_PIXELS = 4
HAMS7031_TOTAL_DARK_PIXELS = 4
HAMS11155_TOTAL_DARK_PIXELS = 20
MIN_ILX_INTTIME = 1.1
MILLI_TO_MICRO = 1000
```

NR\_DIGITAL\_OUTPUTS = 13

NR\_DIGITAL\_INPUTS = 13

NTC1\_ID = 0

NTC2\_ID = 1

TEC\_ID = 2

NR\_ANALOG\_OUTPUTS = 2

ETH\_CONN\_STATUS\_CONNECTING = 0

ETH\_CONN\_STATUS\_CONNECTED = 1

ETH\_CONN\_STATUS\_CONNECTED\_NOMON = 2

ETH\_CONN\_STATUS\_NOCONNECTION = 3

**class DeviceStatus**(value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: `IntEnum`

DeviceStatus enum.

UNKNOWN = 0

USB\_AVAILABLE = 1

USB\_IN\_USE\_BY\_APPLICATION = 2

USB\_IN\_USE\_BY\_OTHER = 3

ETH\_AVAILABLE = 4

ETH\_IN\_USE\_BY\_APPLICATION = 5

ETH\_IN\_USE\_BY\_OTHER = 6

ETH\_ALREADY\_IN\_USE\_USB = 7

**class InterfaceType**(value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: `IntEnum`

InterfaceType enum.

RS232 = 0

USB5216 = 1

USBMINI = 2

USB7010 = 3

ETH7010 = 4

```
class SensType(value, names=None, *, module=None, qualname=None, type=None, start=1,
               boundary=None)
```

Bases: `IntEnum`

SensType enum.

**SENS\_HAMS8378\_256** = 1

**SENS\_HAMS8378\_1024** = 2

**SENS\_ILX554** = 3

**SENS\_HAMS9201** = 4

**SENS\_TCD1304** = 5

**SENS\_TSL1301** = 6

**SENS\_TSL1401** = 7

**SENS\_HAMS8378\_512** = 8

**SENS\_HAMS9840** = 9

**SENS\_ILX511** = 10

**SENS\_HAMS10420\_2048X64** = 11

**SENS\_HAMS11071\_2048X64** = 12

**SENS\_HAMS7031\_1024X122** = 13

**SENS\_HAMS7031\_1024X58** = 14

**SENS\_HAMS11071\_2048X16** = 15

**SENS\_HAMS11155\_2048** = 16

**SENS\_SU256LSB** = 17

**SENS\_SU512LDB** = 18

**SENS\_HAMS11638** = 21

**SENS\_HAMS11639** = 22

**SENS\_HAMS12443** = 23

**SENS\_HAMG9208\_512** = 24

**SENS\_HAMG13913** = 25

**SENS\_HAMS13496** = 26

```
class AvsIdentityType
```

Bases: `Structure`

IdentityType Structure.

**SerialNumber**

Structure/Union member

**Status**

Structure/Union member

**UserFriendlyName**

Structure/Union member

**class BroadcastAnswerType**

Bases: Structure

BroadcastAnswerType Structure.

**InterfaceType**

Structure/Union member

**LocalIp**

Structure/Union member

**RemoteHostIp**

Structure/Union member

**port**

Structure/Union member

**reserved**

Structure/Union member

**serial**

Structure/Union member

**status**

Structure/Union member

**class ControlSettingsType**

Bases: Structure

ControlSettingsType Structure.

**m\_LaserDelay**

Structure/Union member

**m\_LaserWaveLength**

Structure/Union member

**m\_LaserWidth**

Structure/Union member

**m\_StoreToRam**

Structure/Union member

**m\_StrobeControl**

Structure/Union member

**class DarkCorrectionType**

Bases: Structure

DarkCorrectionType Structure.

**m\_Enable**

Structure/Union member

**m\_ForgetPercentage**

Structure/Union member

**class DetectorType**

Bases: Structure

DetectorType Structure.

**m\_DefectivePixels**

Structure/Union member

**m\_ExtOffset**

Structure/Union member

**m\_Gain**

Structure/Union member

**m\_NLEnable**

Structure/Union member

**m\_NrPixels**

Structure/Union member

**m\_Offset**

Structure/Union member

**m\_Reserved**

Structure/Union member

**m\_SensorType**

Structure/Union member

**m\_aFit**

Structure/Union member

**m\_aHighNLCounts**

Structure/Union member

**m\_aLowNLCounts**

Structure/Union member

**m\_aNLCorrect**

Structure/Union member

**class SmoothingType**

Bases: Structure

SmoothingType Structure.

**m\_SmoothModel**  
Structure/Union member

**m\_SmoothPix**  
Structure/Union member

**class SpectrumCalibrationType**  
Bases: Structure  
SpectrumCalibrationType Structure.

**m\_CalInttime**  
Structure/Union member

**m\_Smoothing**  
Structure/Union member

**m\_aCalibConvers**  
Structure/Union member

**class IrradianceType**  
Bases: Structure  
IrradianceType Structure.

**m\_CalibrationType**  
Structure/Union member

**m\_FiberDiameter**  
Structure/Union member

**m\_IntensityCalib**  
Structure/Union member

**class SpectrumCorrectionType**  
Bases: Structure  
SpectrumCorrectionType Structure.

**m\_aSpectrumCorrect**  
Structure/Union member

**class TriggerType**  
Bases: Structure  
TriggerType Structure.

**m\_Mode**  
Structure/Union member

**m\_Source**  
Structure/Union member

**m\_SourceType**  
Structure/Union member

**class MeasConfigType**

Bases: Structure

MeasConfigType Structure.

**m\_Control**

Structure/Union member

**m\_CorDynDark**

Structure/Union member

**m\_IntegrationDelay**

Structure/Union member

**m\_IntegrationTime**

Structure/Union member

**m\_NrAverages**

Structure/Union member

**m\_SaturationDetection**

Structure/Union member

**m\_Smoothing**

Structure/Union member

**m\_StartPixel**

Structure/Union member

**m\_StopPixel**

Structure/Union member

**m\_Trigger**

Structure/Union member

**class TimeStampType**

Bases: Structure

TimeStampType Structure.

**m\_Date**

Structure/Union member

**m\_Time**

Structure/Union member

**class StandAloneType**

Bases: Structure

StandAloneType Structure.

**m\_Enable**

Structure/Union member

**m\_Meas**

Structure/Union member

**m\_Nmsr**

Structure/Union member

**class DynamicStorageType**

Bases: Structure

DynamicStorageType Structure.

**m\_Nmsr**

Structure/Union member

**m\_Reserved**

Structure/Union member

**class TempSensorType**

Bases: Structure

TempSensorType Structure.

**m\_aFit**

Structure/Union member

**class TecControlType**

Bases: Structure

TecControlType Structure.

**m\_Enable**

Structure/Union member

**m\_Setpoint**

Structure/Union member

**m\_aFit**

Structure/Union member

**class ProcessControlType**

Bases: Structure

ProcessControlType Structure.

**m\_AnalogHigh**

Structure/Union member

**m\_AnalogLow**

Structure/Union member

**m\_DigitalHigh**

Structure/Union member

**m\_DigitalLow**

Structure/Union member

**class EthernetSettingsType**

Bases: Structure

EthernetSettingsType Structure.



**m\_DhcpEnabled**  
Structure/Union member

**m\_Gateway**  
Structure/Union member

**m\_IpAddr**  
Structure/Union member

**m\_LinkStatus**  
Structure/Union member

**m\_NetMask**  
Structure/Union member

**m\_TcpPort**  
Structure/Union member

**class OemDataType**  
Bases: Structure  
OemDataType Structure.

**m\_data**  
Structure/Union member

**class HeartbeatRespType**  
Bases: Structure  
HeartbeatRespType Structure.

**m\_BitMatrix**  
Structure/Union member

**m\_Reserved**  
Structure/Union member

**class DeviceConfigType**  
Bases: Structure  
DeviceConfigType Structure.

**m\_ConfigVersion**  
Structure/Union member

**m\_Detector**  
Structure/Union member

**m\_DynamicStorage**  
Structure/Union member

**m\_EthernetSettings**  
Structure/Union member

**m\_Irradiance**  
Structure/Union member

**m\_Len**  
Structure/Union member

**m\_OemData**  
Structure/Union member

**m\_ProcessControl**  
Structure/Union member

**m\_Reflectance**  
Structure/Union member

**m\_SpectrumCorrect**  
Structure/Union member

**m\_StandAlone**  
Structure/Union member

**m\_TecControl**  
Structure/Union member

**m\_aReserved**  
Structure/Union member

**m\_aTemperature**  
Structure/Union member

**m\_aUserFriendlyId**  
Structure/Union member

**activate()**  
Activates the spectrometer for communication.

**Raises**  
        ***AvantesError*** – If there was an error.

**deactivate()**  
Closes communication with the spectrometer.

**get\_analog\_in(*analog\_id*)**  
Get the status of the specified analog input.

**Parameters**  
        **analog\_id** (*int*) – The identifier of the analog input to get.

- AS5216:
  - 0 = thermistor on optical bench (NIR 2.0 / NIR2.2 / NIR 2.5 / TEC)
  - 1 = 1V2
  - 2 = 5VIO
  - 3 = 5VUSB
  - 4 = AI2 = pin 18 at 26-pins connector
  - 5 = AI1 = pin 9 at 26-pins connector
  - 6 = NTC1 onboard thermistor

- 7 = Not used
- Mini:
  - 0 = NTC1 onboard thermistor
  - 1 = Not used
  - 2 = Not used
  - 3 = Not used
  - 4 = AI2 = pin 13 on micro HDMI = pin 11 on HDMI Terminal
  - 5 = AI1 = pin 16 on micro HDMI = pin 17 on HDMI Terminal
  - 6 = Not used
  - 7 = Not used
- AS7010:
  - 0 = thermistor on optical bench (NIR 2.0 / NIR2.2 / NIR 2.5 / TEC)
  - 1 = Not used
  - 2 = Not used
  - 3 = Not used
  - 4 = AI2 = pin 18 at 26-pins connector
  - 5 = AI1 = pin 9 at 26-pins connector
  - 6 = digital temperature sensor, returns degrees Celsius, not Volts
  - 7 = Not used

**Returns**

`float` – The analog input value [Volts or degrees Celsius].

**Raises**

**`AvantesError`** – If there was an error.

**`get_com_port_name()`**

Get the IP address of the device.

**Returns**

`str` – The IP address of the device.

**Raises**

**`AvantesError`** – If there was an error.

**`get_com_type()`**

Get the communication protocol.

**Returns**

`str` – The communication type as defined below:

- RS232 = 0
- USB5216 = 1
- USBMINI = 2

- USB7010 = 3
- ETH7010 = 4
- UNKNOWN = -1

**Raises**

***AvantesError*** – If there was an error.

**get\_dark\_pixel\_data()**

Get the optically black pixel values of the last performed measurement.

You must call `get_data()` before you call this method.

**Returns**

`numpy.ndarray` – The dark pixels.

**Raises**

***AvantesError*** – If there was an error.

**get\_dll\_version()**

Get the DLL version number.

**Returns**

`str` – The DLL version number

**get\_digital\_in(digital\_id)**

Get the status of the specified digital input.

**Parameters**

**digital\_id** (`int`) – The identifier of the digital input to get.

- AS5216:
  - 0 = DI1 = Pin 24 at 26-pins connector
  - 1 = DI2 = Pin 7 at 26-pins connector
  - 2 = DI3 = Pin 16 at 26-pins connector
- Mini:
  - 0 = DI1 = Pin 7 on Micro HDMI = Pin 5 on HDMI terminal
  - 1 = DI2 = Pin 5 on Micro HDMI = Pin 3 on HDMI Terminal
  - 2 = DI3 = Pin 3 on Micro HDMI = Pin 1 on HDMI Terminal
  - 3 = DI4 = Pin 1 on Micro HDMI = Pin 19 on HDMI Terminal
  - 4 = DI5 = Pin 4 on Micro HDMI = Pin 2 on HDMI Terminal
  - 5 = DI6 = Pin 2 on Micro HDMI = Pin 14 on HDMI Terminal
- AS7010:
  - 0 = DI1 = Pin 24 at 26-pins connector
  - 1 = DI2 = Pin 7 at 26-pins connector
  - 2 = DI3 = Pin 16 at 26-pins

**Returns**

`int` – The digital input value.

**Raises**

**AvantesError** – If there was an error.

**get\_handle\_from\_serial**(*serial=None*)

Get the handle ID for the specified serial number.

**Parameters**

**serial** (**str**) – The serial number. Default is to get the status for this object.

**Returns**

**int** – The handle.

**Raises**

**AvantesError** – If there was an error.

**static find**(*path='avaspecx64.dll', port\_id=-1, nmax=16*)

Returns device information for each spectrometer that is connected.

**Parameters**

- **path** (**str**) – The path to the Avantes SDK.
- **port\_id** (**int**) – ID of port to be used. One of:
  - -1: Use both Ethernet (AS7010) and USB ports
  - 0: Use USB port
  - 1..255: Not supported in v9.7 of the SDK
  - 256: Use Ethernet port (AS7010)
- **nmax** (**int**, optional) – The maximum number of devices that can be in the list.

**Returns**

**list** of **AvsIdentityType** – The information about the devices.

**get\_status\_by\_serial**(*serial=None*)

Get the handle ID for the specified serial number.

**Parameters**

**serial** (**str**, optional) – The serial number. Default is to get the status for this object.

**Returns**

**int** – The status.

**Raises**

**AvantesError** – If there was an error.

**done**()

Closes communication and releases internal storage.

**disconnect**()

Closes communication with the spectrometer.

**get\_ip\_config**()

Retrieve IP settings from the spectrometer.

Use this function to read the Ethernet settings of the spectrometer, without having to read the complete device configuration structure.

**Returns**

*EthernetSettingsType* – The Ethernet settings of the spectrometer.

**Raises**

*AvantesError* – If there was an error.

**get\_lambda()**

Returns the wavelength values corresponding to the pixels if available.

**Returns**

*numpy.ndarray* – The wavelength value of each pixel.

**Raises**

*AvantesError* – If there was an error.

**get\_num\_devices()**

Scans for attached devices and returns the number of devices detected.

Deprecated function, replaced by *update\_usb\_devices()*. The functionality is identical.

**Returns**

*int* – The number of devices found.

**get\_oem\_parameter()**

Returns the OEM data structure available on the spectrometer.

**Returns**

*OemDataType* – The OEM parameters.

**Raises**

*AvantesError* – If there was an error.

**get\_parameter()**

Returns the device information of the spectrometer.

**Returns**

*DeviceConfigType* – The device parameters.

**Raises**

*AvantesError* – If there was an error.

**get\_saturated\_pixels()**

Returns, for each pixel, if a pixel was saturated (1) or not (0).

**Returns**

*numpy.ndarray* – The saturation state of each pixel.

**Raises**

*AvantesError* – If there was an error.

**get\_version\_info()**

Returns software version information.

**Returns**

- *str* – FPGA software version.
- *str* – Firmware version.

- `str` – DLL version.

**Raises**

***AvantesError*** – If there was an error.

**get\_data()**

Returns the pixel values of the last performed measurement.

**Returns**

- `int` – Tick count the last pixel of the spectrum was received by the micro-controller. Ticks are in 10 microsecond units since the spectrometer started.
- `numpy.ndarray` – The pixel values.

**Raises**

***AvantesError*** – If there was an error.

**get\_num\_pixels()**

Returns the number of pixels of a spectrometer.

**Raises**

***AvantesError*** – If there was an error.

**heartbeat(*req\_type*)**

Monitor the (heartbeat) functions of the spectrometer.

This function applies only to the AS7010 platform. See the DLL manual for more details.

**Parameters**

**req\_type** (`int`) – The heartbeat request values used to control heartbeat functions.

**Returns**

***HeartbeatRespType*** – The heartbeat response structure received from the spectrometer.

**Raises**

***AvantesError*** – If there was an error.

**init(*port\_id*)**

Initializes the communication interface with the spectrometers and the internal data structures.

For Ethernet devices this function will create a list of available Ethernet spectrometers within all the network interfaces of the host.

**Parameters**

**port\_id** (`int`) – ID of port to be used. One of:

- -1: Use both Ethernet (AS7010) and USB ports
- 0: Use USB port
- 1..255: Not supported in v9.7 of the SDK
- 256: Use Ethernet port (AS7010)

**Returns**

`int` – On success, the number of connected or found devices.

**Raises**

**AvantesError** – If no devices were found.

**measure**(*num\_measurements*, *window\_handle=None*)

Starts measurement on the spectrometer.

**Parameters**

- **num\_measurements** (*int*) – Number of measurements to acquire. Use -1 to measure continuously until `stop_measure()` is called.
- **window\_handle** (*ctypes.c\_void\_p*, optional) – Window handle to notify application measurement result data is available. The DLL sends a message to the window with command: `WM_MEAS_READY`, with `SUCCESS (0)`, the number of scans that were saved in RAM (if `m_StoreToRAM` parameter > 0, see `ControlSettingsType`), or `INVALID_MEAS_DATA` as `WPARAM` value and `a_hDevice` as `LPARAM` value. Set this value to `None` if a callback is not needed.

**Raises**

**AvantesError** – If there was an error.

**measure\_callback**(*num\_measurements*, *callback=None*)

Starts measurement on the spectrometer.

**Parameters**

- **num\_measurements** (*int*) – Number of measurements to acquire. Use -1 to measure continuously until `stop_measure()` is called.
- **callback** (*MeasureCallback*, optional) – A function to notify application measurement result data is available. The DLL will call the given function to notify a measurement is ready and pass two parameters. The first parameter is a reference to the DLL handle. The second parameter is a reference to an integer value: `SUCCESS (0)` if a new scan is available, or the number of scans that were saved in RAM (if `m_StoreToRAM` parameter > 0, see `ControlSettingsType`), or `INVALID_MEAS_DATA (-8)`. Set this value to `None` if a callback is not needed.

**Examples**

```
from msl.equipment.resources.avantes import MeasureCallback

@MeasureCallback
def avantes_callback(handle, info):
    print('The DLL handle is:', handle.contents.value)
    if info.contents.value == 0: # equals 0 if everything is okay
        print(' callback data:', ava.get_data())

# here "ava" is a reference to the AvaSpec class
ava.measure_callback(-1, avantes_callback)
```

**Raises**

**AvantesError** – If there was an error.



**poll\_scan()**

Determines if new measurement results are available.

**Returns**

**int** – Whether there is a scan available: 0 (No) or 1 (Yes).

**Raises**

**AvantesError** – If there was an error.

**prepare\_measure(*config*)**

Prepares measurement on the spectrometer using the specified measurement configuration.

**Parameters**

**config** (*MeasConfigType*) – The measurement configuration.

**Raises**

**AvantesError** – If there was an error.

**register(*handle*)**

Installs an application windows handle to which device attachment/removal messages have to be sent.

**Parameters**

**handle** (*ctypes.c\_void\_p*) – Application window handle.

**Raises**

**AvantesError** – If there was an error.

**reset\_device()**

Performs a hard reset on the given spectrometer.

This function only works with the AS7010 platform.

During reset of the spectrometer, all spectrometer HW modules (microprocessor and USB controller) will be reset at once. The spectrometer will start its reset procedure right after sending the command response back to the host.

**Raises**

**AvantesError** – If there was an error.

**set\_analog\_out(*port\_id*, *value*)**

Sets the analog output value for the specified analog identifier.

**Parameters**

- **port\_id** (*int*) – Identifier for one of the two output signals:
  - AS5216:
    - \* 0 = AO1 = pin 17 at 26-pins connector
    - \* 1 = AO2 = pin 26 at 26-pins connector
  - Mini:
    - \* 0 = AO1 = Pin 12 on Micro HDMI = Pin 10 on HDMI terminal
    - \* 1 = AO2 = Pin 14 on Micro HDMI = Pin 12 on HDMI terminal
  - AS7010:
    - \* 0 = AO1 = pin 17 at 26-pins connector

- \* 1 = AO2 = pin 26 at 26-pins connector

- **value** (*float*) – DAC value to be set in Volts (internally an 8-bits DAC is used) with range 0 - 5.0V.

**Raises**

**AvantesError** – If there was an error.

**set\_digital\_out**(*port\_id*, *value*)

Sets the digital output value for the specified digital identifier.

**Parameters**

- **port\_id** (*int*) – Identifier for one of the 10 output signals:
  - AS5216:
    - \* 0 = DO1 = pin 11 at 26-pins connector
    - \* 1 = DO2 = pin 2 at 26-pins connector
    - \* 2 = DO3 = pin 20 at 26-pins connector
    - \* 3 = DO4 = pin 12 at 26-pins connector
    - \* 4 = DO5 = pin 3 at 26-pins connector
    - \* 5 = DO6 = pin 21 at 26-pins connector
    - \* 6 = DO7 = pin 13 at 26-pins connector
    - \* 7 = DO8 = pin 4 at 26-pins connector
    - \* 8 = DO9 = pin 22 at 26-pins connector
    - \* 9 = DO10 = pin 25 at 26-pins connector
  - Mini:
    - \* 0 = DO1 = Pin 7 on Micro HDMI = Pin 5 on HDMI terminal
    - \* 1 = DO2 = Pin 5 on Micro HDMI = Pin 3 on HDMI Terminal
    - \* 2 = DO3 = Pin 3 on Micro HDMI = Pin 1 on HDMI Terminal
    - \* 3 = DO4 = Pin 1 on Micro HDMI = Pin 19 on HDMI Terminal
    - \* 4 = DO5 = Pin 4 on Micro HDMI = Pin 2 on HDMI Terminal
    - \* 5 = DO6 = Pin 2 on Micro HDMI = Pin 14 on HDMI Terminal
    - \* 6 = Not used
    - \* 7 = Not used
    - \* 8 = Not used
    - \* 9 = Not used
  - AS7010:
    - \* 0 = DO1 = pin 11 at 26-pins connector
    - \* 1 = DO2 = pin 2 at 26-pins connector
    - \* 2 = DO3 = pin 20 at 26-pins connector

- \* 3 = DO4 = pin 12 at 26-pins connector
- \* 4 = DO5 = pin 3 at 26-pins connector
- \* 5 = DO6 = pin 21 at 26-pins connector
- \* 6 = DO7 = pin 13 at 26-pins connector
- \* 7 = DO8 = pin 4 at 26-pins connector
- \* 8 = DO9 = pin 22 at 26-pins connector
- \* 9 = DO10 = pin 25 at 26-pins connector

- **value** (*int*) – The value to be set (0 or 1).

**Raises**

**AvantesError** – If there was an error.

**set\_oem\_parameter**(*parameter*)

Sends the OEM data structure to the spectrometer.

**Parameters**

**parameter** (*OemDataType*) – The OEM data structure.

**Raises**

**AvantesError** – If there was an error.

**set\_parameter**(*parameter*)

Overwrites the device configuration.

Please note that *OemDataType* is part of the DeviceConfigType in EEPROM (see section 3.5 of DLL manual). Precautions must be taken to prevent OEM data overwrites when using *set\_parameter()* method together with *set\_oem\_parameter()*.

**Parameters**

**parameter** (*DeviceConfigType*) – The device parameters.

**Raises**

**AvantesError** – If there was an error.

**set\_prescan\_mode**(*boolean*)

If a prescan is set, the first measurement result will be skipped.

This function is only useful for the AvaSpec-3648 because this detector can be operated in prescan mode, or clear-buffer mode (see DLL manual).

**Parameters**

**boolean** (*bool*) – If *True*, the first measurement result will be skipped (prescan mode), else the detector will be cleared before each new scan (clear-buffer mode).

**Raises**

**AvantesError** – If there was an error.

**set\_pwm\_out**(*port\_id, frequency, duty\_cycle*)

Selects the PWM functionality for the specified digital output.

The PWM functionality is not supported on the Mini.

**Parameters**

- **port\_id** (*int*) – Identifier for one of the 6 PWM output signals:
  - 0 = DO1 = pin 11 at 26-pins connector
  - 1 = DO2 = pin 2 at 26-pins connector
  - 2 = DO3 = pin 20 at 26-pins connector
  - 4 = DO5 = pin 3 at 26-pins connector
  - 5 = DO6 = pin 21 at 26-pins connector
  - 6 = DO7 = pin 13 at 26-pins connector
- **frequency** (*int*) – Desired PWM frequency (500 - 300000) [Hz]. For the AS5216, the frequency of outputs 0, 1 and 2 is the same (the last specified frequency is used) and also the frequency of outputs 4, 5 and 6 is the same. For the AS7010, you can define six different frequencies.
- **duty\_cycle** (*int*) – Percentage high time in one cycle (0 - 100). For the AS5216, channels 0, 1 and 2 have a synchronized rising edge, the same holds for channels 4, 5 and 6. For the AS7010, rising edges are unsynchronized.

**Raises**

**AvantesError** – If there was an error.

**set\_sensitivity\_mode**(*value*)

Set the sensitivity mode.

This method is supported by the following detector types: HAMS9201, HAMG9208\_512, SU256LSB and SU512LDB with the appropriate firmware version.

**Parameters**

**value** (*int*) – 0 for low noise, >0 for high sensitivity

**Raises**

**AvantesError** – If there was an error.

**set\_sync\_mode**(*enable*)

Disables/enables support for synchronous measurement.

**Parameters**

**enable** (*bool*) – **False** to disable sync mode, **True** to enable sync mode.

**Raises**

**AvantesError** – If there was an error.

**stop\_measure**()

Stops the measurement.

**Raises**

**AvantesError** – If there was an error.

**suppress\_stray\_light**(*factor*)

Returns the stray light corrected pixel values of a dark corrected measurement.

**Parameters**

**factor** (*float*) – Multiplication factor for the stray light algorithm.

**Returns**

- `numpy.ndarray` – Scope minus dark.
- `numpy.ndarray` – Stray light suppressed.

**Raises**

**`AvantesError`** – If there was an error.

**`update_eth_devices(nmax=16)`**

Return the number of Ethernet devices that are connected to the computer.

Internally checks the list of connected Ethernet devices and returns the number of devices attached. If the `properties` attribute contains a key 'port\_id' with a value of -1 then the returned value also includes the number of USB devices.

**Parameters**

**`nmax`** (`int`, optional) – The maximum number of devices that can be found.

**Returns**

`list` of `BroadcastAnswerType` – The information about the devices.

**`update_usb_devices()`**

Return the number of USB devices that are connected to the computer.

Internally checks the list of connected USB devices and returns the number of devices attached. If the `properties` attribute contains a key 'port\_id' with a value of -1 then the returned value also includes the number of Ethernet devices.

**Returns**

`int` – The number of devices found.

**`use_high_res_adc(enable)`**

Enable the 16-bit AD converter.

When using the 16 bit ADC in full High Resolution mode (0.65535), please note that the irradiance intensity calibration, as well as the nonlinearity calibration are based on the 14bit ADC range. Therefore, if using the nonlinearity correction or irradiance calibration in your own software using the High Resolution mode, you need to apply the additional correction with ADCFactor (= 4.0), as explained in detail in section 4.6.1 and 4.6.3 of the manual.

**Parameters**

**`enable`** (`bool`) – If `True` use a 16-bit AD converter, otherwise use a 14-bit ADC.

**Raises**

**`AvantesError`** – If there was an error.

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

Resources for equipment from [Bentham](#).

## Submodules

### `msl.equipment.resources.bentham.benhw32` module

A wrapper around the 32-bit Bentham `benhw32_cdec1` SDK.

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

Bases: `Server32`

A wrapper around the 32-bit Bentham `benhw32_cdec1` SDK.

Do not instantiate this class directly.

The SDK is provided for 32-bit Windows only. This module is run on a 32-bit `Server` so that the `Bentham` class can be run on a 64-bit Python interpreter in order to access the functions in the SDK from a 64-bit process.

`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 [Bentham32](#) class.

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

Bases: [Connection](#)

A wrapper around the [Bentham32](#) class.

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

The [properties](#) for a Bentham connection supports the following key-value pairs in the [Connections Database](#):

`'cfg': str`, the path to the `System.cfg` file [default: **None**]  
`'atr': str`, the path to the `System.atr` file [default: **None**]

If the `cfg` and `atr` values are not defined in the [Connections Database](#) then you will have to call [build\\_system\\_model\(\)](#), [load\\_setup\(\)](#) and [initialise\(\)](#) (in that order) to configure the SDK.

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

### **Parameters**

**record** ([EquipmentRecord](#)) – A record from an [Equipment-Register Database](#).

**auto\_measure()**

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

Set the model configuration file.

### **Parameters**

**path** ([str](#)) – The path to the `System.cfg` file.

**disconnect()**

Disconnect from the SDK and from the 32-bit server.

**errcheck(result, \*args, \*\*kwargs)**

Checks whether a function call to the SDK was successful.

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

**get\_component\_list()**

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



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

**property wavelength**

**initialise**()

Initialize the connection.

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

Load the setup file.

**Parameters**

**path** (*str*) – The path to the `System.atr` file.

**park**()

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

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

**version**()

*str*: The version number of the SDK.

**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.cmi package**

Resources for equipment from [CMI](#).

### **Submodules**

## **msl.equipment.resources.cmi.sia3 module**

Establishes a connection to the Switched Integrator Amplifier (SIA3 board) that is designed by the [Czech Metrology Institute](#).

```
class msl.equipment.resources.cmi.sia3.IntegrationTime(value, names=None, *,  
                                                    module=None,  
                                                    qualname=None, type=None,  
                                                    start=1, boundary=None)
```

Bases: [IntEnum](#)

The amount of time to integrate the photo-diode signal.

`TIME_50u = 5`

`TIME_100u = 6`

`TIME_1m = 7`

`TIME_10m = 8`

`TIME_20m = 9`

`TIME_100m = 10`

`TIME_200m = 11`

`TIME_500m = 12`

`TIME_1 = 13`

`TIME_2 = 14`

**class** `msl.equipment.resources.cmi.sia3.SIA3(record)`

Bases: [`ConnectionSerial`](#)

Establishes a connection to the Switched Integrator Amplifier (SIA3 board) that is designed by the [Czech Metrology Institute](#).

Do not instantiate this class directly. Use the [`connect\(\)`](#) method to connect to the equipment.

**Parameters**

**record** ([`EquipmentRecord`](#)) – A record from an [`Equipment-Register Database`](#).

**GAIN**

The gain (i.e., the integration time)

alias of [`IntegrationTime`](#)

**set\_integration\_time(time)**

Set the integration time (i.e., the gain).

**Parameters**

**time** ([`IntegrationTime`](#)) – The integration time as a [`IntegrationTime`](#) enum value or member name, e.g., `sia.set_integration_time('10m')`, `sia.set_integration_time(sia.GAIN.TIME_10m)` and `sia.set_integration_time(8)` are all equivalent statements.

**set\_ps(ps)**

Set the timer pre-scale value.

The timer pre-scale value divides the microprocessor internal frequency by something similar to  $2^{\text{PS}}$ . Therefore, to reach a 2-second integration time the *ps* value must be set to the maximum value of 7.

**Parameters**

**ps** ([`int`](#)) – The timer pre-scale value. Must be in the range [0, 7].

**Raises**

[`ValueError`](#) – If the value of *ps* is invalid.

## msl.equipment.resources.dataray package

Resources for equipment from [DataRay](#).

### Submodules

#### msl.equipment.resources.dataray.datarayocx\_32 module

Load the 32-bit DATARAYOCX library using [MSL-LoadLib](#).

**class** `msl.equipment.resources.dataray.datarayocx_32.DataRayOCX32`(*host, port, \*\*kwargs*)

Bases: [Server32](#)

Communicates with the 32-bit DATARAYOCX library.

Tested with the WinCamD-LCM-8.0D36 software version.

**wait\_to\_configure()**

Wait until the camera has been configured.

**capture**(*timeout*)

Capture an image.

**start()**

Start the camera.

**stop()**

Stop the camera.

**shutdown\_handler()**

Stop the camera and close the application.

#### msl.equipment.resources.dataray.datarayocx\_64 module

Establishes a connection to the DATARAYOCX library developed by DataRay Inc.

**class** `msl.equipment.resources.dataray.datarayocx_64.DataRayOCX64`(*record*)

Bases: [Connection](#)

A wrapper around the [DataRayOCX32](#) class.

This class can be used with either a 32- or 64-bit Python interpreter to call the 32-bit functions in the DATARAYOCX library. A GUI is created to configure and visualize the images taken by the camera.

Tested with the WinCamD-LCM-8.0D36 software version.

The [properties](#) for a DataRay connection supports the following key-value pairs in the [Connections Database](#):

```
'area_filter': int, area filter: 1=1pixel, 2=3pixels, 3=5pixels, 4=7pixels, 5=9pixels [default: 1]
'camera_index': int, the camera to use (between 0 and 7; 0=first camera found) [default: 0]
'centroid_method': int, the centroid method to use (0, 1 or 2) [default: 0]
'filter': float, percent full scale filter (0, 0.1, 0.2, 0.5, 1, 2, 5 or 10) [default: 0.2]
'major_minor_method': int, the major/minor method to use (0, 1 or 2) [default: 0]
'ui_size': int, the size of the User Interface (value=height of a button in pixels) [default: 25]
```

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

#### `wait_to_configure()`

Wait until the camera has been configured.

This is a blocking call and waits until you close the popup Window.

#### `capture(timeout=10, restart=False)`

Capture an image.

#### Parameters

- **timeout** (*float*, optional) – The maximum number of seconds to wait to capture an image.
- **restart** (*bool*, optional) – Whether to keep the camera running after the image is captured.

#### Returns

*dict* – The information about the captured image. The key-value pairs are:

- **adc\_peak\_**: *float*, the maximum ADC value (as a percentage)
- **area\_filter**: *int*, area filtering applies a convolution to the pixels (1=1pixel, 2=3pixels, 3=5pixels, 4=7pixels, 5=9pixels)
- **centroid**: *tuple*, the (x, y) centroid value
- **centroid\_filter\_size**: *int*, centroid filter size (in pixels)
- **centroid\_type**: *int*, the centroid type (0, 1 or 2)
- **clip\_level\_centroid**: *float*, the centroid clip level (between 0 and 1)
- **clip\_level\_geo**: *float*, the geometric clip level (between 0 and 1)
- **crosshair**: *float*, the angle between the horizontal x-axis and the solid crosshair line (in degrees)
- **eff\_2w**: *float*, the effective beam size (in um)

- `effective_centroid`: `tuple`, the effective (x, y) centroid value
- `effective_geo_centroid`: `tuple`, the effective (x, y) geometric centroid value
- `ellip`: `float`, the ratio between the minor/major axis
- `elp`: `float`, the beam azimuthal angle for ISO 11146 (in degrees)
- `exposure_time`: `float`, the exposure time (in ms)
- `filter_full_scale`: `float`, used in the triangular weighting smoothing function
- `image`: `numpy.ndarray`, the camera image
- `image_zoom`: `float`, the zoom factor of the image
- `imager_gain`: `float`, the gain used by the imager
- `inc_p`: `float`, Dxx power (in Watts)
- `inc_power_area`: `float`, Dxx beam area (in mm<sup>2</sup>)
- `inc_power_major`: `float`, Dxx beam diameter along the major axis (in mm)
- `inc_power_mean`: `float`, Dxx mean beam diameter (in mm)
- `inc_power_minor`: `float`, Dxx beam diameter along the minor axis (in mm)
- `is_fast_update`: `bool`, whether the camera is in fast update or normal mode
- `is_full_resolution`: `bool`, whether the camera is set to full resolution
- `maj_iso`: `float`, the ISO 11146 beam size along the major axis (in um)
- `major`: `float`, the beam size along the major axis (in um)
- `major_minor_method`: `int`, the major/minor method used (0, 1 or 2)
- `mean`: `float`, the mean beam size (in um)
- `mean_theta`: `float`, Dxx mean angle (in mrad)
- `min_iso`: `float`, the ISO 11146 beam size along the minor axis (in um)
- `minor`: `float`, the beam size along the minor axis (in um)
- `num_averages`: `int`, the number of averages per capture
- `num_captures`: `int`, the number of images that have been captured
- `orient`: `float`, the angle between the horizontal x-axis and the major or minor axis closest to the horizontal x-axis (in degrees)
- `peak`: `tuple`, the peak location in (x, y), usually zero
- `pixel_width_um`: `float`, the width of a pixel (in um)
- `pixel_height_um`: `float`, the height of a pixel (in um)

- `pk_to_avg`: `float`, the peak to average value
- `plateau_uniformity`: `float`, the flatness of the plateau (between 0 and 1)
- `profile_x`: `numpy.ndarray`, the profile in the X direction
- `profile_y`: `numpy.ndarray`, the profile in the Y direction
- `roi`: `tuple`, the selected region of interest (x, y, width, height)
- `xc`: `float`, the centroid position along the x axis (in um)
- `xg`: `float`, the geometric centroid position along the x axis (in um)
- `xp`: `float`, the peak-intensity centroid position along the x axis (in um)
- `xu`: `float`, the user-selected centroid position along the x axis (in um)
- `yc`: `float`, the centroid position along the y axis (in um)
- `yg`: `float`, the geometric centroid position along the y axis (in um)
- `yp`: `float`, the peak-intensity centroid position along the y axis (in um)
- `yu`: `float`, the user-selected centroid position along the y axis (in um)

**Raises**

***DataRayError*** – If not successful.

**disconnect()**

Disconnect from the camera.

**start()**

Start the camera.

**stop()**

Stop the camera.

## **msl.equipment.resources.electron\_dynamics package**

Resources for equipment from *Electron Dynamics*.

### **Submodules**

#### **msl.equipment.resources.electron\_dynamics.tc\_series module**

Establishes a connection to a TC Series Temperature Controller from Electron Dynamics Ltd.

**class** `msl.equipment.resources.electron_dynamics.tc_series.TCSeries`(*record*)

Bases: *ConnectionSerial*

Establishes a connection to a TC Series Temperature Controller from Electron Dynamics Ltd.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**get\_alarm()**

Get the alarm parameters.

**Returns**

*list* – The alarm parameters. See *set\_alarm()* for more details.

**get\_control()**

Get the control parameters.

**Returns**

*list* – The control parameters. See *set\_control()* for more details.

**get\_output()**

Get the output parameters.

**Returns**

*list* – The output parameters. See *set\_output()* for more details.

**get\_sensor()**

Get the sensor parameters.

**Returns**

*list* – The sensor parameters. See *set\_sensor()* for more details.

**get\_setpoint()**

Get the setpoint parameters.

**Returns**

*list* – The setpoint parameters. See *set\_setpoint()* for more details.

**get\_status()**

Get the status.

**Returns**

*list* – The status parameters.

**set\_alarm(alarm\_type, minimum, maximum, ok\_min, ok\_max, limit\_min, limit\_max)**

Set the alarm parameters.

This corresponds to the *c* command in the Commands manual.

**Parameters**

- **alarm\_type** (*int*) – The alarm type. One of:
  - 0: None
  - 1: Minimum
  - 2: Maximum
  - 3: Both
- **minimum** (*float*) – Sets the temperature below which the alarm is activated.

- **maximum** (*float*) – Sets the temperature above which the alarm is activated.
- **ok\_min** (*float*) – Sets the lower temperature difference point from the setpoint for temperature OK.
- **ok\_max** (*float*) – Sets the higher temperature difference point from the setpoint for temperature OK.
- **limit\_min** (*float*) – Sets the temperature minimum, below which the drive output is disabled.
- **limit\_max** (*float*) – Sets the temperature maximum, above which the drive output is disabled.

**set\_control**(*control\_type, p, i, d, d\_filter, dead\_band, power\_up\_state*)

Set the control type and the control parameters.

This corresponds to the a command in the Commands manual.

#### Parameters

- **control\_type** (*int*) – The control type. One of:
  - 0: None
  - 1: On/Off - Output drive is only fully On (heating or cooling) or Off
  - 2: Proportional (P)
  - 3: Proportional and Integral (PI)
  - 4: Proportional, Integral and Derivative (PID)
- **p** (*float*) – The proportional (gain) value. With proportional action, the controller output is proportional to the temperature error from the setpoint. The proportional terms sets the gain for this where:  $\text{Output} = (\text{setpoint} - \text{actual temperature}) * \text{proportional term}$
- **i** (*float*) – The integral value. With integral action, the controller output is proportional to the amount of time the error is present. Integral action eliminates offset. The integral term is a time unit in seconds. NB for larger effects of integration reduce the integral time, also for operation without integral, integral time can be set to a large number e.g. 1e6.
- **d** (*float*) – The derivative value. With derivative action, the controller output is proportional to the rate of change of the measurement or error. The controller output is calculated by the rate of change of the measurement with time, in seconds. To increase the derivative action increase the derivative value. See also Derivative Filter (*d\_filter*).
- **d\_filter** (*float*) – The derivative filter is a low pass filter function on the derivative value. This allows the filtration of noise components which are a problem with a pure derivative function. The filter value should be set to be between 0 and 1.
- **dead\_band** (*float*) – For use with On/Off control the dead band specifies the temperature range around the set point where the output is zero. For example:



- Temperature > setpoint + dead\_band (Fully Cooling)
- Temperature < setpoint - dead\_band (Fully Heating)
- Temperature < setpoint + dead\_band AND > setpoint-dead\_band (Output off)
- **power\_up\_state** (*int*) – This sets the temperature control state from power up. One of:
  - 0: Off
  - 1: On
  - 2: Same as the last setting prior to power off

**set\_output**(*polarity, minimum, maximum, frequency*)

Set the output parameters.

This corresponds to the g command in the Commands manual.

#### Parameters

- **polarity** (*int*) – Sets the polarity of the output drive. One of:
  - 0: Negative
  - 1: Positive
- **minimum** (*float*) – Sets the minimum value limit of the output. Range -100 to +100.
- **maximum** (*float*) – Sets the maximum value limit of the output. Range -100 to +100.
- **frequency** (*float*) – Sets the pulse-width modulation repetition frequency of the output drive. Range 20 to 1000 Hz.

**set\_output\_drive**(*mode, value*)

Set the output drive state and value.

This corresponds to the m command in the Commands manual.

#### Parameters

- **mode** (*int*) – The drive mode. Either 0 (Off) or 1 (On).
- **value** (*int*) – Percent output.

**set\_sensor**(*sensor, x2, x, c, unit, averaging, rl=22000*)

Set the sensor type and the sensor parameters.

This corresponds to the e command in the Commands manual.

#### Parameters

- **sensor** (*int*) – The sensor type. One of:
  - 0: Voltage
  - 1: PT100
  - 2: LM35
  - 3: LM50

- 4: LM60
  - 5: LM61
  - 6: NTC Thermistor
  - 7: RES
  - 8: PT1000
  - 9: RTD
  - **x2** (*float*) –
  - **x** (*float*) –
  - **c** (*float*) – The  $x_2$ ,  $x$  and  $c$  parameters are quadratic coefficients that can be used to convert the sensor voltage into a temperature, i.e.,  $\text{temperature} = (v * v * x_2) + (v * x) + c$ , where  $v$  is the measured sensor voltage.
- For NTC thermistors  $x_2$  is the beta value as specified for the thermistor type,  $x$  is the resistance at 25 deg C, and,  $c$  is still the offset.
- **unit** (*str*) – The temperature units. One of:
    - 'C': Centigrade
    - 'F': Fahrenheit
    - 'K': Kelvin
    - 'V': Voltage
    - 'R': Resistance
  - **averaging** (*int*) – Set the averaging to be 0 (Off) or 1 (On).
  - **rl** (*float*, optional) – Used for NTC or RES sensors. This value corresponds to the RL drive resistance.

**set\_setpoint**(*method, value, pot\_range, pot\_offset*)

Set the setpoint parameters.

This corresponds to the **i** command in the Commands manual.

#### Parameters

- **method** (*int*) – The temperature setpoint can be set via software or by altering the potentiometer on the temperature controller hardware. One of:
  - 0: Potentiometer
  - 1: Software
  - 2: Input
- **value** (*float*) – The setpoint value.
- **pot\_range** (*float*) – The temperature range of the potentiometer.
- **pot\_offset** (*float*) – The minimum temperature point of the potentiometer.

**set\_test**(*mode, arg1, arg2, arg3, arg4, arg5, arg6, arg7*)

Set the test parameters.

This corresponds to the **k** command in the Commands manual.

#### Parameters

- **mode** (*int*) – The test mode. One of:
  - 0: Off
  - 1: Normal
  - 2: Temperature cycle
  - 3: Temperature ramp
  - 4: Auto tune
- **arg1** (*float*) – The maximum cycle value or the test time (in seconds).
- **arg2** (*float*) – The start temperature or setpoint value.
- **arg3** (*float*) – The end temperature or end peak test value.
- **arg4** (*float*) – The rate 1 or the calc PID value.
- **arg5** (*float*) – The rate 2 or the run PID value.
- **arg6** (*float*) – The time 1 or undo PID value.
- **arg7** (*float*) – The time 2 or auto cal value.

### msl.equipment.resources.energetiq package

Resources for equipment from [Energetiq](#).

#### Submodules

### msl.equipment.resources.energetiq.eq99 module

Communicate with the EQ-99 Manager from Energetiq.

**class** `msl.equipment.resources.energetiq.eq99.EQ99`(*record*)

Bases: [ConnectionSerial](#)

Communicate with the EQ-99 Manager from Energetiq.

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**identity**()

Query the instrument identification.

**Returns**

**str** – Returns the identification string for the instrument in the following format: *Energetiq Model SN Ver Build*

**reset()**

Resets the instrument to factory defaults and the output is shut off.

The unit remains in remote mode.

**get\_beep()**

Query whether beeps are enabled.

**Returns**

**bool** – Whether beeps are enabled.

**set\_beep(*beep*=2)**

Set the beep value.

**Parameters**

**beep** (**int** or **bool**, optional) – Causes the instrument to beep, or enables or disabled the beep sound for error messages and other events that generate and audible response. Possible values are

- 0 or **False** – Disable the beep sound
- 1 or **True** – Enable the beep sound
- 2 – Generate one beep

**get\_brightness()**

Query the display brightness.

**Returns**

**int** – Returns the value of the display brightness (between 0 and 100).

**set\_brightness(*brightness*)**

Set the display brightness.

**Parameters**

**brightness** (**int**) – Sets the display brightness level from 0 to 100 percent. There are only 8 brightness levels (each separated by about 12.5 percent) and the brightness value is used to select an appropriate level.

**delay(*milliseconds*)**

Specify a delay to use in command processing.

**Parameters**

**milliseconds** (**int**) – Causes command processing to be delayed for the specified number of milliseconds. Valid range is from 1 to 30000 milliseconds.

**condition\_register()**

Query LDLS condition register.

The condition register reflects the state of the instrument at the time the condition register is read.

The bitmask sequence is as follows:

| Index | Value | Description             |
|-------|-------|-------------------------|
| 0     | 1     | Interlock               |
| 1     | 2     | Controller not detected |
| 2     | 4     | Controller fault        |
| 3     | 8     | Lamp fault              |
| 4     | 16    | Output on               |
| 5     | 32    | Lamp on                 |
| 6     | 64    | Laser on                |
| 7     | 128   | Laser stable            |
| 8     | 256   | Shutter open            |

### Returns

- `int` – The condition register value.
- `str` – The condition register as a bitmask string. For example, a value of 336 is expressed as '000010101' (meaning that the interlock is closed, there are no faults, the output is on, the lamp is off, the laser is on, the laser is not stable and the shutter is open).

### `event_register()`

Query LDLS event register.

Returns the LDLS event register. The event register reflects the occurrence of any condition since the last time the event register was read. For example, if the output was turned on and then turned off, the Output on the bit in the condition register will be zero, but the same bit in the event register will be one.

The bitmask sequence is as follows:

| Index | Value | Description             |
|-------|-------|-------------------------|
| 0     | 1     | Interlock               |
| 1     | 2     | Controller not detected |
| 2     | 4     | Controller fault        |
| 3     | 8     | Lamp fault              |
| 4     | 16    | Output on               |
| 5     | 32    | Lamp on                 |
| 6     | 64    | Laser on                |
| 7     | 128   | Laser stable            |
| 8     | 256   | Shutter open            |

### Returns

- `int` – The event register value.
- `str` – The event register as a bitmask string. For example, a value of 256 is expressed as '000000001' and 32 as '000001000'.

### `get_exposure_time()`

Query the exposure time.

**Returns**

`int` – The exposure time, in milliseconds.

**set\_exposure\_time**(*milliseconds*)

Set the exposure time.

Exposure time is used when the shutter exposure mode is set to *Exposure mode* (see [`set\_exposure\_mode\(\)`](#)). An exposure is triggered by a shutter button press or the shutter trigger input.

**Parameters**

**milliseconds** (`int`) – The exposure time, in milliseconds, from 100 to 30000 ms.

**get\_exposure\_mode**()

Query the exposure mode.

**Returns**

`int` – The exposure mode (0=Manual, 1=Exposure).

**set\_exposure\_mode**(*mode*)

Set the exposure mode.

Same as the Shutter setting in the menu.

**Parameters**

**mode** (`int` or `bool`) – The exposure mode (0=Manual, 1=Exposure).

**get\_output**()

Query the output state.

**Returns**

`bool` – Whether the output is enabled.

**set\_output**(*enable*)

Turn the output on or off.

**Parameters**

**enable** (`int` or `bool`) – Whether to enable the output.

**get\_lamptime**()

Query the lamp runtime.

**Returns**

`float` – The number of hours accumulated while the lamp was on.

**set\_lamptime**(*hours*)

Set the lamp runtime.

Resets the runtime to the new value. Useful for resetting the runtime to zero when the lamp has been serviced or replaced, or when moving the manager to a new LDLS system.

**Parameters**

**hours** (`float`) – The lamp runtime, in hours, between 0 and 9999.

**get\_shutter\_init**()

Query the power-up shutter state.

**Returns**

`int` – The power-up shutter state.

- 0 – Shutter is closed on power-up
- 1 – Shutter is open on power-up

**set\_shutter\_init(*state*)**

Set the power-up shutter state

**Parameters**

**state** (`int` or `bool`) – Sets the initial state of the shutter on power-up of the manager

- 0 or `False` – Shutter is closed on power-up
- 1 or `True` – Shutter is open on power-up

**get\_shutter\_state()**

Query the shutter state.

**Returns**

`bool` – The state of the shutter.

- `False` – Shutter is closed
- `True` – Shutter is open

**set\_shutter\_state(*state*)**

Open, close, or trigger the shutter.

A close command (state equals 0) will always close the shutter, regardless of exposure mode. An open command (state equals 1) will open the shutter if exposure mode is set to Manual, or trigger a shutter if exposure mode is set to Exposure.

**Parameters**

**state** (`int` or `bool`) – The state of the shutter.

- 0 or `False` – Close the shutter
- 1 or `True` – Open or trigger the shutter

**get\_trigger\_mode()**

Query the trigger mode.

**Returns**

`int` – The trigger mode. See [`set\_trigger\_mode\(\)`](#) for more details.

**set\_trigger\_mode(*mode*)**

Set the trigger mode.

The trigger mode controls how the shutter trigger input controls the operation of the shutter. For more information on trigger modes, see *Shutter Operation* in the *Operating the Instrument* section of the manual for more details.

**Parameters**

**mode** (`int`) – The trigger mode.

- 0 – Positive edge trigger

- 1 – Negative edge trigger
- 2 – Positive level trigger
- 3 – Negative level trigger
- 4 – Off (trigger disabled)

#### **get\_message\_buffer()**

Query the internal message buffer.

##### **Returns**

**str** – The value of the internal message buffer.

#### **set\_message\_buffer(message)**

Set the message buffer.

##### **Parameters**

**message** (**str**) – Sets the internal message buffer, up to a maximum of 16 characters. If more than 16 characters are specified then the additional characters are silently ignored.

#### **get\_remote\_mode\_error()**

Query whether errors are displayed while in remote mode.

##### **Returns**

**bool** – Whether errors are displayed while in remote mode.

#### **set\_remote\_mode\_error(enable)**

Set whether to display errors while in remote mode.

This command controls if the instrument will display errors while in remote mode. If set to zero, then errors will not be displayed. If set to one, errors will be displayed. Errors will always accumulate in the error queue.

##### **Parameters**

**enable** (**int** or **bool**) – Whether to display errors while in remote mode.

- 0 or **False** – No not display errors in remote mode
- 1 or **True** – Display errors in remote mode

#### **serial\_number()**

Query the serial number of the instrument.

##### **Returns**

**str** – The serial number of the instrument. This is the same information that is part of the \*IDN? query.

#### **get\_termination()**

Query response terminator.

Returns the current response terminator setting. See [set\\_termination\(\)](#) for a complete definition of possible return values.

##### **Returns**

**int** – The response terminator.



**set\_termination(value)**

Set the response terminator character(s).

This command controls the termination characters used for responses to queries.

**Parameters**

**value** (*int*) – The response terminator character(s)

- 0 or 1 – <CR><LF>
- 2 or 3 – <CR>
- 4 or 5 – <LF>
- 6 or 7 – no terminator

**run\_time()**

Query run time.

**Returns**

*str* – Returns the elapsed time since the unit has been turned on. Format is in HH:MM:SS.ss, where HH is hours, MM is minutes, SS is seconds, and ss is hundredths of a second.

**timer()**

Query time since the last time this method was called.

**Returns**

*str* – Returns the elapsed time since the last time this method was called, or, if this is the first time calling this method then the time since unit has been turned on. Format is in HH:MM:SS.ss, where HH is hours, MM is minutes, SS is seconds, and ss is hundredths of a second.

**version()**

Query the firmware version.

**Returns**

*str* – Returns the firmware version. This is the same information that is part of the \*IDN? query.

## **msl.equipment.resources.mks\_instruments package**

Resources for equipment from MKS Instruments.

### **Submodules**

#### **msl.equipment.resources.mks\_instruments.pr4000b module**

Flow and Pressure controller, PR4000B, from MKS Instruments.

**class** `msl.equipment.resources.mks_instruments.pr4000b.PR4000B(record)`

Bases: *ConnectionSerial*

Flow and Pressure controller, PR4000B, from MKS Instruments.

The default settings for the RS232 connection are:

- Baud rate = 9600
- Data bits = 7
- Stop bits = 1
- Parity = ODD
- Flow control = None

The baud rate and parity can be changed on the controller. The data bits, stop bits, and flow control cannot be changed. A null modem (cross over) cable is required when using a USB to RS232 converter. RS485 support is not implemented.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

```
ERROR_CODES = {'#E001': 'Communication Error', '#E002': 'ADC Overflow or Underflow', '#E003': 'Range Error, Setpoint < 0 or out of range', '#E010': 'Syntax Error', '#E020': 'Failed to execute command', '#W001': 'Offset > 250 mV'}
```

```
UNITS = {0: 'ubar', 1: 'mbar', 2: 'bar', 3: 'mTor', 4: 'Torr', 5: 'KTor', 6: 'Pa', 7: 'kPa', 8: 'mH2O', 9: 'cH2O', 10: 'PSI', 11: 'N/qm', 12: 'SCCM', 13: 'SLM', 14: 'SCM', 15: 'SCFH', 16: 'SCFM', 17: 'mA', 18: 'V', 19: '%', 20: 'C'}
```

```
SIGNAL_MODES = {0: 'METER', 1: 'OFF', 2: 'INDEP', 3: 'EXTRN', 4: 'SLAVE', 5: 'RTD'}
```

```
LIMIT_MODES = {0: 'SLEEP', 1: 'LIMIT', 2: 'BAND', 3: 'MLIMIT', 4: 'MBAND'}
```

```
TAGS = {0: 'SP', 1: 'VA', 2: 'CH', 3: 'FL', 4: 'PR', 5: 'EX'}
```

**auto\_zero**(*channel*)

Auto zero a channel

#### Parameters

**channel** (*int*) – The channel, either 1 or 2.

#### Returns

*int* – The offset.

**default**(*mode*)

Reset to the default configuration.

#### Parameters

**mode** (*str*) – The mode to reset. One of Pressure, Flow, P or F (case insensitive).

**displays\_enable**(*display, enable*)

Turn a display on or off.

#### Parameters

- **display** (*int*) – The display number [1, 4].
- **enable** (*bool*) – Whether to turn the display on, *True*, or off, *False*.

**displays\_setup**(*display*, *line*, *tag*, *channel*)

Configure a display.

**Parameters**

- **display** (*int*) – The display number [1, 4].
- **line** (*int*) – The line number, 1 or 2.
- **tag** (*int* or *str*) – The tag to use (0=SP, 1=VA, 2=CH, 3=FL, 4=PR, 5=EX). For example, setting tag to 4 or 'PR' are equivalent.
- **channel** (*int*) – The channel, either 1 or 2.

**display\_4**(*enable*)

Whether to enable or disable display 4.

**Parameters**

- **enable** (*bool*) – Whether to enable or disable display 4.

**external\_input**(*channel*)

Return the external input of a channel

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.

**Returns**

- *float* – The external input.

**get\_access\_channel**(*channel*)

Get the setpoint and the state of the valve of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.

**Returns**

- *float* – The setpoint value.
- *bool* – Whether the valve is on, *True*, or off, *False*.

**get\_actual\_value**(*channel*)

Get the actual value of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.

**Returns**

- *float* – The value.

**get\_address**()

Get the address.

**Returns**

- *int* – The address.

**get\_dead\_band**(*channel*)

Get the dead band of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*float* – The dead band.

**get\_dialog**()

Get the current dialog index that is displayed.

**Returns**

*int* – The dialog index.

**get\_display\_text**()

Get the display text.

**Returns**

*str* – The display text.

**get\_external\_input\_range**(*channel*)

Get the external input range of a channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*int* – The external input range.

**get\_external\_output\_range**(*channel*)

Get the external output range of a channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*int* – The external output range.

**get\_formula\_relay**(*channel*)

Get the relay formula of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*str* – The formula.

**get\_formula\_temporary**(*channel*)

Get the temporary formula of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*str* – The formula.

**get\_gain**(*channel*)

Get the gain of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*float* – The gain.

**get\_input\_range(channel)**

Get the input range of a channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*int* – The input range.

**get\_interface\_mode()**

Get the interface mode.

**Returns**

*int* – The interface mode.

**get\_limit\_mode(channel)**

Get the limit mode of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

- *int* – The index of the limit mode.
- *str* – The name of the limit mode.

**get\_linearization\_point(channel, point)**

Get the point in the linearization table of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **point** (*int*) – The point in the table [0, 10].

**Returns**

- *float* – The x value.
- *float* – The y value.

**get\_linearization\_size(channel)**

Get the size of the linearization table of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*int* – The size of the table.

**get\_lower\_limit(channel)**

Get the lower limit of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

`float` – The lower limit.

**get\_offset(*channel*)**

Get the offset of a particular channel.

**Parameters**

**channel** (`int`) – The channel, either 1 or 2.

**Returns**

`int` – The offset.

**get\_output\_range(*channel*)**

Get the output range of a channel.

**Parameters**

**channel** (`int`) – The channel, either 1 or 2.

**Returns**

`int` – The output range.

**get\_range(*channel*)**

Get the range and unit of a channel.

**Parameters**

**channel** (`int`) – The channel, either 1 or 2.

**Returns**

- `float` – The range.
- `int` – The unit index.
- `str` – The unit name.

**get\_relays(*channel*)**

Get the relay state of a channel.

**Parameters**

**channel** (`int`) – The channel, either 1 or 2.

**Returns**

`bool` – Whether the relay is enabled or disabled.

**get\_remote\_mode()**

Get the remote operation mode.

**Returns**

`bool` – Whether the remote operation mode is enabled, `True`, or disabled, `False`.

**get\_resolution()**

Get whether 16-bit resolution is enabled.

**Returns**

`bool` – Whether 16-bit resolution is enabled, `True`, or disabled, `False`.

**get\_rtd\_offset(*channel*)**

Get the RTD offset of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*int* – The offset.

**get\_scale(channel)**

Get the scale of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*float* – The scale.

**get\_setpoint(channel)**

Get the setpoint of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*float* – The setpoint.

**get\_signal\_mode(channel)**

Get the signal mode of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

- *int* – The index number of the signal mode.
- *str* – The name of the signal mode.

**get\_upper\_limit(channel)**

Get the upper limit of a particular channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*float* – The upper limit.

**get\_valves(channel)**

Get the state of the valve of a channel.

**Parameters**

**channel** (*int*) – The channel, either 1 or 2.

**Returns**

*bool* – Whether the valve is enabled or disabled.

**identity()**

Returns the identity.

**Returns**

*str* – The identity (e.g., PR42vvrrsssss, where vv is the version, rr is the release and ssss is the serial number).

### **lock()**

Lock setup.

### **request\_key()**

Requests most recent key that was pressed.

#### **Returns**

- **int** – The key that was most recently pressed.
- **int** – The number of key presses that occurred since the last time this method was called.

### **reset\_status()**

Send the reset/status command.

### **set\_access\_channel(channel, setpoint, valve)**

Set the setpoint and the state of the valve for a particular channel.

#### **Parameters**

- **channel** (**int**) – The channel, either 1 or 2.
- **setpoint** (**float**) – The setpoint value.
- **valve** (**bool**) – Whether to enable or disable the valve.

#### **Returns**

**float** – The actual setpoint value.

### **set\_actual\_value(channel, setpoint)**

Set the actual value of a particular channel.

#### **Parameters**

- **channel** (**int**) – The channel, either 1 or 2.
- **setpoint** (**float**) – The setpoint.

#### **Returns**

**float** – The actual value.

### **set\_address(address)**

Set the address.

#### **Parameters**

**address** (**int**) – The address [0, 31].

### **set\_dead\_band(channel, band)**

Set the dead band of a particular channel.

#### **Parameters**

- **channel** (**int**) – The channel, either 1 or 2.
- **band** (**float**) – The dead band [0.0% to 9.9% of full scale].

### **set\_dialog(index)**

Set the display dialog.



**Parameters**

**index** (*int*) – The dialog index (between 0 and 29 inclusive). See Appendix D of the manual for more information.

**set\_display\_text**(*text*, *clear=True*)

Set the display text.

To view the text on the display you must call `set_dialog()` with the index equal to 3.

**Parameters**

- **text** (*str*) – The text to display. Maximum 32 characters.
- **clear** (*bool*, optional) – Whether to clear the current display text before setting the new text.

**set\_external\_input\_range**(*channel*, *range*)

Set the external input range of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **range** (*int*) – The external input range [1, 10] in Volts.

**set\_external\_output\_range**(*channel*, *range*)

Set the external output range of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **range** (*int*) – The external output range [1, 10] in Volts.

**set\_formula\_relay**(*channel*, *formula*)

Set the relay formula of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **formula** (*str*) – The relay formula.

**set\_formula\_temporary**(*channel*, *formula*)

Set the temporary formula of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **formula** (*str*) – The temporary formula.

**set\_gain**(*channel*, *gain*)

Set the gain of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **gain** (*float*) – The gain [0.001, 2.000].

**set\_input\_range**(*channel*, *range*)

Set the input range of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **range** (*int*) – The input range [1, 10] in Volts.

**set\_interface\_mode**(*mode*)

Set the interface mode.

**Parameters**

- **mode** (*int*) – The interface mode.

**set\_limit\_mode**(*channel*, *mode*)

Set the limit mode of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **mode** (*int* or *str*) – The limit mode as either an index number [0, 4] or a name (e.g., SLEEP).

**set\_linearization\_point**(*channel*, *point*, *x*, *y*)

Set a point in the linearization table of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **point** (*int*) – The point in the table [0, 10].
- **x** (*float*) – The x value [-5% to 100% of full scale].
- **y** (*float*) – The y value [-5% to 100% of full scale].

**set\_linearization\_size**(*channel*, *size*)

Set the size of the linearization table of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **size** (*int*) – The size of the table.

**set\_lower\_limit**(*channel*, *limit*)

Set the lower limit of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **limit** (*float*) – The lower limit [-5% to 110% of full scale].

**set\_offset**(*channel*, *offset*)

Set the offset of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.

- **offset** (*int*) – The offset [-250, 250].

**set\_output\_range**(*channel, range*)

Set the output range of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **range** (*int*) – The output range [1, 10] in Volts.

**set\_range**(*channel, range, unit*)

Set the range and unit of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **range** (*float*) – The range value.
- **unit** (*int* or *str*) – The unit as either an index number [0, 20] or a name (e.g., kPa).

**set\_relays**(*channel, enable*)

Set the relay state of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **enable** (*bool*) – Whether to enable or disable the relay.

**set\_remote\_mode**(*enable*)

Set the remote operation mode to be enable or disabled.

**Parameters**

- **enable** (*bool*) – Whether to enable or disable remote operation.

**set\_resolution**(*enable*)

Set the 16-bit resolution to be enable or disabled.

**Parameters**

- **enable** (*bool*) – Whether to enable or disable 16-bit resolution.

**set\_rtd\_offset**(*channel, offset*)

Set the RTD offset of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **offset** (*int*) – The RTD offset [-250, 250].

**set\_scale**(*channel, scale*)

Set the scale of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **scale** (*float*) – The scale.

**set\_setpoint**(*channel*, *setpoint*)

Set the setpoint of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **setpoint** (*float*) – The setpoint.

**set\_signal\_mode**(*channel*, *mode*)

Set the range and unit of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **mode** (*int* or *str*) –  
The signal mode as either an index number (e.g., between 0 and 5 inclusive)  
or a name (e.g., INDEP).

**set\_tweak\_control**(*enable*)

Set tweak control.

**Parameters**

- **enable** (*bool*) – Whether to switch tweak control on or off.

**set\_upper\_limit**(*channel*, *limit*)

Set the upper limit of a particular channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.
- **limit** (*float*) – The upper limit [-5% to 110% of full scale].

**set\_valves**(*channel*, *enable*)

Set the state of the valve of a channel.

**Parameters**

- **channel** (*int*) – The channel, either 1 or 2.

**status**()

Request status bits.

**Returns**

- *int* – The status value.
- *str* – The binary representation of the value.

**unlock**()

Unlock setup.

## **msl.equipment.resources.nkt package**

Resources for equipment from [NKT Photonics](#).

### **Submodules**

#### **msl.equipment.resources.nkt.nktpdll module**

Wrapper around the NKTPDLL.dll SDK from NKT Photonics.

The wrapper was written using v2.1.2.766 of the SDK.

##### **msl.equipment.resources.nkt.nktpdll.PortStatusCallback**

Use as a decorator for a callback function when a port status changes.

##### **msl.equipment.resources.nkt.nktpdll.DeviceStatusCallback**

Use as a decorator for a callback function when a device status changes.

##### **msl.equipment.resources.nkt.nktpdll.RegisterStatusCallback**

Use as a decorator for a callback function when a register status changes.

##### **class msl.equipment.resources.nkt.nktpdll.DateTimeType**

Bases: `Structure`

The DateTimeType struct (24 hour format).

###### **Day**

Structure/Union member

###### **Hour**

Structure/Union member

###### **Min**

Structure/Union member

###### **Month**

Structure/Union member

###### **Sec**

Structure/Union member

###### **Year**

Structure/Union member

##### **class msl.equipment.resources.nkt.nktpdll.ParameterSetType**

Bases: `Structure`

The ParameterSet struct.

This is how calculation on parameter sets is done internally by modules:

$\text{DAC\_value} = (\text{value} * (\text{X/Y})) + \text{Offset}$

where, value is either `ParameterSetType::StartVal` or `ParameterSetType::FactoryVal`

$\text{value} = (\text{ADC\_value} * (\text{X/Y})) + \text{Offset}$

where, value often is available via another measurement register.

**Denominator**

Structure/Union member

**ErrorHandler**

Structure/Union member

**FactoryVal**

Structure/Union member

**LLimit**

Structure/Union member

**Numerator**

Structure/Union member

**Offset**

Structure/Union member

**StartVal**

Structure/Union member

**ULimit**

Structure/Union member

**Unit**

Structure/Union member

```
class msl.equipment.resources.nkt.nktpdll.DeviceModeTypes(value, names=None, *,
                                                         module=None,
                                                         qualname=None,
                                                         type=None, start=1,
                                                         boundary=None)
```

Bases: `IntEnum`

The DeviceModeTypes enum.

**DevModeDisabled = 0**

**DevModeAnalyzeInit = 1**

**DevModeAnalyze = 2**

**DevModeNormal = 3**

**DevModeLogDownload = 4**

**DevModeError = 5**

**DevModeTimeout = 6**

**DevModeUpload = 7**

```
class msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes(value, names=None, *,
                                                         module=None,
                                                         qualname=None,
                                                         type=None, start=1,
                                                         boundary=None)
```

Bases: `IntEnum`

The DeviceStatusTypes enum.

**DeviceModeChanged** = 0

**DeviceLiveChanged** = 1

**DeviceTypeChanged** = 2

**DevicePartNumberChanged** = 3

**DevicePCBVersionChanged** = 4

**DeviceStatusBitsChanged** = 5

**DeviceErrorCodeChanged** = 6

**DeviceBlVerChanged** = 7

**DeviceFwVerChanged** = 8

**DeviceModuleSerialChanged** = 9

**DevicePCBSerialChanged** = 10

**DeviceSysTypeChanged** = 11

```
class msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes(value, names=None, *,
                                                            module=None,
                                                            qualname=None,
                                                            type=None, start=1,
                                                            boundary=None)
```

Bases: `IntEnum`

The ParamSetUnitTypes enum

**UnitNone** = 0

**UnitmV** = 1

**UnitV** = 2

**UnituA** = 3

**UnitmA** = 4

**UnitA** = 5

**UnituW** = 6

**UnitcmW** = 7

UnitdmW = 8  
UnitmW = 9  
UnitW = 10  
UnitmC = 11  
UnitcC = 12  
UnitdC = 13  
Unitpm = 14  
Unitdm = 15  
Unitm = 16  
UnitPerCent = 17  
UnitPerMille = 18  
UnitcmA = 19  
UnitdmA = 20  
UnitRPM = 21  
UnitdBm = 22  
UnitcBm = 23  
UnitmBm = 24  
UnitdB = 25  
UnitcB = 26  
UnitmB = 27  
Unitdpm = 28  
UnitcV = 29  
UnitdV = 30  
Unitlm = 31  
Unitdlm = 32  
Unitclm = 33  
Unitmlm = 34



```
class msl.equipment.resources.nkt.nktpdll.RegisterPriorityTypes(value,
                                                                names=None, *,
                                                                module=None,
                                                                qualname=None,
                                                                type=None,
                                                                start=1,
                                                                boundary=None)
```

Bases: `IntEnum`

The RegisterPriorityTypes enum.

**RegPriority\_Low = 0**

**RegPriority\_High = 1**

```
class msl.equipment.resources.nkt.nktpdll.RegisterDataTypes(value, names=None, *,
                                                            module=None,
                                                            qualname=None,
                                                            type=None, start=1,
                                                            boundary=None)
```

Bases: `IntEnum`

The RegisterDataTypes enum.

**RegData\_Unknown = 0**

**RegData\_Mixed = 1**

**RegData\_U8 = 2**

**RegData\_S8 = 3**

**RegData\_U16 = 4**

**RegData\_S16 = 5**

**RegData\_U32 = 6**

**RegData\_S32 = 7**

**RegData\_F32 = 8**

**RegData\_U64 = 9**

**RegData\_S64 = 10**

**RegData\_F64 = 11**

**RegData\_Ascii = 12**

**RegData\_Paramset = 13**

**RegData\_B8 = 14**

**RegData\_H8 = 15**

**RegData\_B16 = 16**

**RegData\_H16** = 17

**RegData\_B32** = 18

**RegData\_H32** = 19

**RegData\_B64** = 20

**RegData\_H64** = 21

**RegData\_DateTime** = 22

```
class msl.equipment.resources.nkt.nktpdll.RegisterStatusTypes(value, names=None,
                                                                *, module=None,
                                                                qualname=None,
                                                                type=None, start=1,
                                                                boundary=None)
```

Bases: `IntEnum`

The RegisterStatusTypes enum.

**RegSuccess** = 0

**RegBusy** = 1

**RegNacked** = 2

**RegCRCErr** = 3

**RegTimeout** = 4

**RegComError** = 5

```
class msl.equipment.resources.nkt.nktpdll.PortStatusTypes(value, names=None, *,
                                                            module=None,
                                                            qualname=None,
                                                            type=None, start=1,
                                                            boundary=None)
```

Bases: `IntEnum`

The PortStatusTypes enum

**PortStatusUnknown** = 0

**PortOpening** = 1

**PortOpened** = 2

**PortOpenFail** = 3

**PortScanStarted** = 4

**PortScanProgress** = 5

**PortScanDeviceFound** = 6

**PortScanEnded** = 7

**PortClosing** = 8

**PortClosed** = 9

**PortReady** = 10

**class** msl.equipment.resources.nkt.nktpdll.**NKT**(*record*)

Bases: [Connection](#)

Wrapper around the NKTPDLL.dll SDK from NKT Photonics.

The *properties* for a NKT connection supports the following key-value pairs in the [Connections Database](#):

'sdk\_path': `str`, The path to the SDK [default: 'NKTPDLL.dll']  
'open\_port': `bool`, Whether to automatically `open` the port [default: `True`]  
'auto': `bool`, Whether to `open` the port `with` bus scanning [default: `True`]  
'live': `bool`, Whether to `open` the port `in` live mode [default: `True`]

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

#### Parameters

**record** ([EquipmentRecord](#)) – A record from an [Equipment-Register Database](#).

#### RegisterStatusCallback

alias of CFunctionType

#### PortStatusCallback

alias of CFunctionType

#### DeviceStatusCallback

alias of CFunctionType

**class** DeviceModeTypes(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: [IntEnum](#)

The DeviceModeTypes enum.

**DevModeDisabled** = 0

**DevModeAnalyzeInit** = 1

**DevModeAnalyze** = 2

**DevModeNormal** = 3

**DevModeLogDownload** = 4

**DevModeError** = 5

**DevModeTimeout** = 6

**DevModeUpload** = 7

```
class DeviceStatusTypes(value, names=None, *, module=None, qualname=None,  
                        type=None, start=1, boundary=None)
```

Bases: `IntEnum`

The DeviceStatusTypes enum.

**DeviceModeChanged** = 0

**DeviceLiveChanged** = 1

**DeviceTypeChanged** = 2

**DevicePartNumberChanged** = 3

**DevicePCBVersionChanged** = 4

**DeviceStatusBitsChanged** = 5

**DeviceErrorCodeChanged** = 6

**DeviceBlVerChanged** = 7

**DeviceFwVerChanged** = 8

**DeviceModuleSerialChanged** = 9

**DevicePCBSerialChanged** = 10

**DeviceSysTypeChanged** = 11

```
class ParamSetUnitTypes(value, names=None, *, module=None, qualname=None,  
                        type=None, start=1, boundary=None)
```

Bases: `IntEnum`

The ParamSetUnitTypes enum

**UnitNone** = 0

**UnitmV** = 1

**UnitV** = 2

**UnituA** = 3

**UnitmA** = 4

**UnitA** = 5

**UnituW** = 6

**UnitcmW** = 7

**UnitdmW** = 8

**UnitmW** = 9

**UnitW** = 10

```
UnitmC = 11
UnitcC = 12
UnitdC = 13
Unitpm = 14
Unitdnm = 15
Unitnm = 16
UnitPerCent = 17
UnitPerMille = 18
UnitcmA = 19
UnitdmA = 20
UnitRPM = 21
UnitdBm = 22
UnitcBm = 23
UnitmBm = 24
UnitdB = 25
UnitcB = 26
UnitmB = 27
Unitdpm = 28
UnitcV = 29
UnitdV = 30
Unitlm = 31
Unitdlm = 32
Unitclm = 33
Unitmlm = 34
```

```
class RegisterPriorityTypes(value, names=None, *, module=None, qualname=None,
                           type=None, start=1, boundary=None)
```

```
Bases: IntEnum
```

```
The RegisterPriorityTypes enum.
```

```
RegPriority_Low = 0
```

```
RegPriority_High = 1
```

```
class RegisterDataTypes(value, names=None, *, module=None, qualname=None,  
                        type=None, start=1, boundary=None)
```

Bases: `IntEnum`

The RegisterDataTypes enum.

**RegData\_Unknown** = 0

**RegData\_Mixed** = 1

**RegData\_U8** = 2

**RegData\_S8** = 3

**RegData\_U16** = 4

**RegData\_S16** = 5

**RegData\_U32** = 6

**RegData\_S32** = 7

**RegData\_F32** = 8

**RegData\_U64** = 9

**RegData\_S64** = 10

**RegData\_F64** = 11

**RegData\_Ascii** = 12

**RegData\_Paramset** = 13

**RegData\_B8** = 14

**RegData\_H8** = 15

**RegData\_B16** = 16

**RegData\_H16** = 17

**RegData\_B32** = 18

**RegData\_H32** = 19

**RegData\_B64** = 20

**RegData\_H64** = 21

**RegData\_DateTime** = 22

```
class RegisterStatusTypes(value, names=None, *, module=None, qualname=None,  
                        type=None, start=1, boundary=None)
```

Bases: `IntEnum`

The RegisterStatusTypes enum.

**RegSuccess** = 0

**RegBusy** = 1

**RegNacked** = 2

**RegCRCErr** = 3

**RegTimeout** = 4

**RegComError** = 5

**class PortStatusTypes**(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: [IntEnum](#)

The PortStatusTypes enum

**PortStatusUnknown** = 0

**PortOpening** = 1

**PortOpened** = 2

**PortOpenFail** = 3

**PortScanStarted** = 4

**PortScanProgress** = 5

**PortScanDeviceFound** = 6

**PortScanEnded** = 7

**PortClosing** = 8

**PortClosed** = 9

**PortReady** = 10

**static close\_ports**(*names=None*)

Close the specified port name(s).

**Parameters**

**names** (*str*, *list* of *str*, optional) – If *None* then close all opened ports. If a *str* then the name of a port. Otherwise a *list* of names. Port names are case sensitive.

**Raises**

[NKTErrors](#) – If there was an error calling this method.

**static load\_sdk**(*path=None*)

Load the SDK.

**Parameters**

**path** (*str*, optional) – The path to NKTPDLL.dll. If not specified then searches for the library.

**static device\_get\_all\_types**(*opened\_ports=None, size=255*)

Returns all device types (module types) from the internal device list.

**Parameters**

- **opened\_ports** (*str* or *list* of *str*, optional) – A port or a list of opened ports. If not specified then the *get\_open\_ports()* method will be called and the types for each port will be returned.
- **size** (*int*, optional) – The maximum number of bytes that the device list can be.

**Returns**

*dict* – The port names are the keys and each value is *dict* with the module type as the keys and its corresponding device ID as the value.

**Raises**

*NKTEError* – If there was an error calling this method.

**device\_create**(*device\_id, wait\_ready*)

Creates a device in the internal device list.

If the *open\_ports()* function has been called with *live = 1* then the kernel immediately starts to monitor the device.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **wait\_ready** (*bool*) – *False* - Don't wait for the device to be ready. *True* - Wait up to 2 seconds for the device to complete its analyze cycle. (All standard registers being successfully read)

**Raises**

*NKTEError* – If there was an error calling this method.

**device\_exists**(*device\_id*)

Checks if a specific device already exists in the internal device list.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*bool* – Whether the device exists.

**Raises**

*NKTEError* – If there was an error calling this method.

**device\_get\_boot\_loader\_version**(*device\_id*)

Returns the boot-loader version (*int*) for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*int* – The boot-loader version.



**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_boot\_loader\_version\_str(device\_id)**

Returns the boot-loader version (string) for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*str* – The boot-loader version.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_error\_code(device\_id)**

Returns the error code for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*int* – The error code.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_firmware\_version(device\_id)**

Returns the firmware version (int) for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*int* – The firmware version.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_firmware\_version\_str(device\_id)**

Returns the firmware version (string) for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*str* – The firmware version.

**Raises**

*NKError* – If there was an error calling this method.

**device\_get\_live(device\_id)**

Returns the internal device live status for a specific device id.

Requires the port being already opened with the *open\_ports()* function and the device being already created, either automatically or with the *device\_create()* function.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*bool* – Whether live mode is enabled.

**Raises**

*NKError* – If there was an error calling this method.

**device\_get\_mode(device\_id)**

Returns the internal device mode for a specific device id.

Requires the port being already opened with the *open\_ports()* function and the device being already created, either automatically or with the *device\_create()* function.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*DeviceModeTypes* – The device mode type.

**Raises**

*NKError* – If there was an error calling this method.

**device\_get\_module\_serial\_number\_str(device\_id)**

Returns the module serial number (string) for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*str* – The serial number.

**Raises**

*NKError* – If there was an error calling this method.

**device\_get\_part\_number\_str(device\_id)**

Returns the part number for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*str* – The part number.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_pcb\_serial\_number\_str(*device\_id*)**

Returns the PCB serial number (string) for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*str* – The part number.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_pcb\_version(*device\_id*)**

Returns the PCB version for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*int* – The PCB version number.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_status\_bits(*device\_id*)**

Returns the status bits for a given device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*int* – The status bits.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_get\_type(*device\_id*)**

Returns the module type for a specific device id.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*int* – The module type.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_remove**(*device\_id*)

Remove a specific device from the internal device list.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Raises**

***NKError*** – If there was an error calling this method.

**device\_remove\_all**()

Remove all devices from the internal device list.

No confirmation is given, the list is simply cleared.

**Raises**

***NKError*** – If there was an error calling this method.

**device\_set\_live**(*device\_id*, *live\_mode*)

Sets the internal device live status for a specific device id (module address).

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **live\_mode** (*bool*) – Set to **True** to enable.

**Raises**

***NKError*** – If there was an error calling this method.

**disconnect**()

Disconnect from the port.

**Raises**

***NKError*** – If there was an error calling this method.

**static get\_all\_ports**(*size=255*)

Returns a list of all ports.

**Parameters**

**size** (*int*, optional) – The maximum size of the string buffer to fetch the results.

**Returns**

*list* of *str* – A list of port names.

**get\_modules**(*size=255*)

Returns all device types (module types) from the device.

**Parameters**

**size** (*int*, optional) – The maximum number of bytes that the device list can be.

**Returns**

*dict* – The module type as the keys and its corresponding device ID as the value.

**Raises**

***NKTError*** – If there was an error calling this method.

**static get\_legacy\_bus\_scanning()**

Get the bus-scanning mode.

**Returns**

**bool** – **True** if in legacy mode otherwise in normal mode.

**static get\_open\_ports(size=255)**

Returns a list of already-opened ports.

**Parameters**

**size** (**int**, optional) – The maximum size of the string buffer to fetch the results.

**Returns**

**list** of **str** – A list of port names that are already open.

**get\_port\_error\_msg()**

Retrieve error message for the port.

**Returns**

**str** – The error message. An empty string indicates no error.

**Raises**

***NKTError*** – If there was an error calling this method.

**get\_port\_status()**

Get the status of the port.

**Returns**

***PortStatusTypes*** – The port status.

**Raises**

***NKTError*** – If there was an error calling this method.

**static open\_ports(names=None, auto=True, live=True)**

Open the specified port(s).

Repeated calls to this function is allowed to reopen and/or rescan for devices.

**Parameters**

- **names** (**str**, **list** of **str**, optional) – If **None** then open all available ports. If a **str** then the name of a port. Otherwise a **list** of names. Port names are case sensitive. Example port names are 'AcoustikPort1', 'COM6'.
- **auto** (**bool**, optional) – If **True** then automatically start bus scanning and add the found devices in the internal device list. If **False** then bus scanning and device creation is not automatically handled. The port is automatically closed if no devices are found.
- **live** (**bool**, optional) – If **True** then keep all the found or created devices in live mode, which means the Interbus kernel keeps monitoring all the found devices and their registers. Please note that this will keep the modules watchdog alive as long as the port is open. If **False** then disable continuous monitoring of the registers. No callback is possible

on register changes. Use the `register_read()`, `register_write()` and `register_write_read()` methods.

**Raises**

**`NKError`** – If there was an error calling this method.

**`point_to_point_port_add`**(*host\_address*, *host\_port*, *client\_address*, *client\_port*, *protocol*, *ms\_timeout=100*)

Creates or modifies a point to point port.

**Parameters**

- **`host_address`** (`str`) – The local ip address, e.g., '192.168.1.67'.
- **`host_port`** (`int`) – The local port number.
- **`client_address`** (`str`) – The remote ip address, e.g., '192.168.1.100'.
- **`client_port`** (`int`) – The remote port number.
- **`protocol`** (`int`) – Either 0 (TCP) or 1 (UDP).
- **`ms_timeout`** (`int`, optional) – Telegram timeout value in milliseconds.

**Raises**

**`NKError`** – If there was an error calling this method.

**`point_to_point_port_del`**()

Delete the point-to-point port.

**Raises**

**`NKError`** – If there was an error calling this method.

**`point_to_point_port_get`**()

Retrieve the information about the point-to-point port setting.

**Returns**

`dict` – The information about the point-to-point port setting.

**Raises**

**`NKError`** – If there was an error calling this method.

**`register_read`**(*device\_id*, *reg\_id*, *index=-1*)

Reads a register value and returns the result.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **`device_id`** (`int`) – The device id (module address).
- **`reg_id`** (`int`) – The register id (register address).
- **`index`** (`int`, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

`bytes` – The register value.

**Raises**

***NKTError*** – If there was an error calling this method.

**register\_create**(*device\_id, reg\_id, priority, data\_type*)

Creates a register in the internal register list.

If the **open\_ports()** function has been called with the *live* = 1 then the kernel immediately starts to monitor the register.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **priority** (*int*) – The *RegisterPriorityTypes* (monitoring priority).
- **data\_type** (*int*) – The *RegisterDataTypes*, not used internally but could be used in a common callback function to determine data type.

**Raises**

***NKTError*** – If there was an error calling this method.

**register\_exists**(*device\_id, reg\_id*)

Checks if a specific register already exists in the internal register list.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).

**Returns**

*bool* – Whether the register exists.

**Raises**

***NKTError*** – If there was an error calling this method.

**register\_get\_all**(*device\_id*)

Returns the register ids (register addresses) from the internal register list.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Returns**

*list* of *int* – The register ids.

**Raises**

***NKTError*** – If there was an error calling this method.

**register\_read\_ascii**(*device\_id, reg\_id, index=-1*)

Reads an ascii string from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).

- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*str* – The ascii value.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_read\_f32**(*device\_id*, *reg\_id*, *index=-1*)

Reads 32-bit float value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*float* – The 32-bit float value.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_read\_f64**(*device\_id*, *reg\_id*, *index=-1*)

Reads 64-bit double value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*float* – The 64-bit double value.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_read\_s16**(*device\_id*, *reg\_id*, *index=-1*)

Reads 16-bit signed short value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).



- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 16-bit signed short value.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_read\_s32**(*device\_id*, *reg\_id*, *index=-1*)

Reads 32-bit signed long value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 32-bit signed long value.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_read\_s64**(*device\_id*, *reg\_id*, *index=-1*)

Reads 64-bit signed long long value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 64-bit signed long long value.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_read\_s8**(*device\_id*, *reg\_id*, *index=-1*)

Reads 8-bit signed char value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).

- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 8-bit signed char value.

**Raises**

***NKTErrors*** – If there was an error calling this method.

**register\_read\_u16**(*device\_id*, *reg\_id*, *index=-1*)

Reads 16-bit unsigned short value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 16-bit unsigned short value.

**Raises**

***NKTErrors*** – If there was an error calling this method.

**register\_read\_u32**(*device\_id*, *reg\_id*, *index=-1*)

Reads 32-bit unsigned long value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 32-bit unsigned long value.

**Raises**

***NKTErrors*** – If there was an error calling this method.

**register\_read\_u64**(*device\_id*, *reg\_id*, *index=-1*)

Reads 64-bit unsigned long long value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).

- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 64-bit unsigned long long value.

**Raises**

***NKError*** – If there was an error calling this method.

**register\_read\_u8**(*device\_id*, *reg\_id*, *index=-1*)

Reads 8-bit unsigned char value from the register.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **index** (*int*, optional) – Value index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted.

**Returns**

*int* – The 8-bit unsigned char value.

**Raises**

***NKError*** – If there was an error calling this method.

**register\_remove**(*device\_id*, *reg\_id*)

Remove a specific register from the internal register list.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).

**Raises**

***NKError*** – If there was an error calling this method.

**register\_remove\_all**(*device\_id*)

Remove all registers from the internal register list.

No confirmation given, the list is simply cleared.

**Parameters**

**device\_id** (*int*) – The device id (module address).

**Raises**

***NKError*** – If there was an error calling this method.

**register\_write**(*device\_id*, *reg\_id*, *data*, *index=-1*)

Writes a register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).

- **reg\_id** (*int*) – The register id (register address).
- **data** (*bytes*) – The data to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

***NKError*** – If there was an error calling this method.

**register\_write\_ascii**(*device\_id, reg\_id, string, write\_eol, index=-1*)

Writes a string to the register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **string** (*str*) – The string to write to the register.
- **write\_eol** (*bool*) – Whether to append the End Of Line character (a null character) to the string.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a mixed-type register.

**Raises**

***NKError*** – If there was an error calling this method.

**register\_write\_f32**(*device\_id, reg\_id, value, index=-1*)

Writes a 32-bit float register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*float*) – The 32-bit float to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

***NKError*** – If there was an error calling this method.

**register\_write\_f64**(*device\_id, reg\_id, value, index=-1*)

Writes a 64-bit double register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).

- **reg\_id** (*int*) – The register id (register address).
- **value** (*float*) – The 64-bit double to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read**(*device\_id, reg\_id, data, index=-1*)

Writes then reads a register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **data** (*bytes*) – The data to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*bytes* – The data that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_ascii**(*device\_id, reg\_id, string, write\_eol, index=-1*)

Writes then reads a string register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **string** (*str*) – The string to write to the register.
- **write\_eol** (*bool*) – Whether to append the End Of Line character (a null character) to the string.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*str* – The string that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_f32**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 32-bit float register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*float*) – The 32-bit float value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*float* – The 32-bit float value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_f64**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 64-bit double register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*float*) – The 64-bit double value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*float* – The 64-bit double value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_s16**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 16-bit signed short register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 16-bit signed short value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

`int` – The 16-bit signed short value that was written to the register.

**Raises**

**`NKError`** – If there was an error calling this method.

**`register_write_read_s32`**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 32-bit signed long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **`device_id`** (`int`) – The device id (module address).
- **`reg_id`** (`int`) – The register id (register address).
- **`value`** (`int`) – The 32-bit signed long value to write to the register.
- **`index`** (`int`, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

`int` – The 32-bit signed long value that was written to the register.

**Raises**

**`NKError`** – If there was an error calling this method.

**`register_write_read_s64`**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 64-bit signed long long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **`device_id`** (`int`) – The device id (module address).
- **`reg_id`** (`int`) – The register id (register address).
- **`value`** (`int`) – The 64-bit signed long long value to write to the register.
- **`index`** (`int`, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

`int` – The 64-bit signed long long value that was written to the register.

**Raises**

**`NKError`** – If there was an error calling this method.

**`register_write_read_s8`**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 8-bit signed char register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **`device_id`** (`int`) – The device id (module address).
- **`reg_id`** (`int`) – The register id (register address).

- **value** (*int*) – The 8-bit signed char value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*int* – The 8-bit signed char value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_u16**(*device\_id, reg\_id, value, index=-1*)

Writes then reads a 16-bit unsigned short register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 16-bit unsigned short value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*int* – The 16-bit unsigned short value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_u32**(*device\_id, reg\_id, value, index=-1*)

Writes then reads a 32-bit unsigned long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 32-bit unsigned long value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*int* – The 32-bit unsigned long value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_u64**(*device\_id, reg\_id, value, index=-1*)

Writes then reads a 64-bit unsigned long long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.



**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 64-bit unsigned long long value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*int* – The 64-bit unsigned long long value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_read\_u8**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes then reads a 8-bit unsigned char register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 8-bit unsigned char value to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Returns**

*int* – The 8-bit unsigned char value that was written to the register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_s16**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 16-bit signed short register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 16-bit signed short to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

*NKTError* – If there was an error calling this method.

**register\_write\_s32**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 32-bit signed long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 32-bit signed long to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

**NKTErrors** – If there was an error calling this method.

**register\_write\_s64**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 64-bit signed long long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 64-bit signed long long to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

**NKTErrors** – If there was an error calling this method.

**register\_write\_s8**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 8-bit signed char register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 8-bit signed char to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

**NKTErrors** – If there was an error calling this method.

**register\_write\_u16**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 16-bit unsigned short register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 16-bit unsigned short to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

**NKTErrors** – If there was an error calling this method.

**register\_write\_u32**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 32-bit unsigned long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 32-bit unsigned long to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

**NKTErrors** – If there was an error calling this method.

**register\_write\_u64**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 64-bit unsigned long long register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

**Parameters**

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 64-bit unsigned long long to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

**Raises**

**NKTErrors** – If there was an error calling this method.

**register\_write\_u8**(*device\_id*, *reg\_id*, *value*, *index=-1*)

Writes a 8-bit unsigned char register value.

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

#### Parameters

- **device\_id** (*int*) – The device id (module address).
- **reg\_id** (*int*) – The register id (register address).
- **value** (*int*) – The 8-bit unsigned char to write to the register.
- **index** (*int*, optional) – Value index. Typically -1, but could be used to write a value in a multi-value register.

#### Raises

**NKTErrors** – If there was an error calling this method.

**static set\_legacy\_bus\_scanning**(*mode*)

Set the bus-scanning mode to normal or legacy.

#### Parameters

**mode** (*bool*) – If **False** then bus scanning is set to normal mode and allows for a rolling masterId. In this mode the masterId is changed for each message to allow for out-of-sync detection. If **True** then bus scanning is set to legacy mode and fixes the masterId at address 66(0x42). Some older modules do not accept masterIds other than 66(0x42).

**static set\_callback\_device\_status**(*callback*)

Enables/Disables a callback for device status changes.

#### Parameters

**callback** (*DeviceStatusCallback*) – A *DeviceStatusCallback* object. Pass in **None** to disable callbacks.

---

**Note:** Due to a risk of circular runaway leading to stack overflow, it is not allowed to call functions in the DLL from within the callback function. If a call is made to a function in the DLL the function will raise an exception.

---

## Examples

```
from ctypes import c_ubyte
from msl.equipment.resources import NKT

@NKT.DeviceStatusCallback
def device_callback(port, dev_id, status, length, address):
    # 'address' is an integer and represents the address of c_void_p_
    ↪from the callback
    data = bytearray((c_ubyte * length).from_address(address)[:])
    print('The port is {!r}'.format(port))
    print('The device ID is {}'.format(dev_id))
```

(continues on next page)

(continued from previous page)

```

    print('The device status is {!r}'.format(NKT.
→DeviceStatusTypes(status)))
    print('The device data is {!r}'.format(data))

NKT.set_callback_device_status(device_callback)

```

**static set\_callback\_port\_status(*callback*)**

Enables/Disables a callback for port status changes.

Used by the *open\_ports()* and *close\_ports()* functions.

**Parameters**

**callback** (*PortStatusCallback*) – A *PortStatusCallback* object.  
Pass in *None* to disable callbacks.

---

**Note:** Due to a risk of circular runaway leading to stack overflow, it is not allowed to call functions in the DLL from within the callback function. If a call is made to a function in the DLL the function will raise an exception.

---

## Examples

```

from msl.equipment.resources import NKT

@NKT.PortStatusCallback
def port_callback(port, status, cur_scan, max_scan, device):
    print('The port is {!r}'.format(port))
    print('The port status is {!r}'.format(NKT.
→PortStatusTypes(status)))
    print('Current scanned address or device found address is {}'.
→format(cur_scan))
    print('There are {} addresses to scan in total'.format(max_scan))
    print('Found device with type {}'.format(device))

NKT.set_callback_port_status(port_callback)

```

**static set\_callback\_register\_status(*callback*)**

Enables/Disables a callback for register status changes.

**Parameters**

**callback** (*RegisterStatusCallback*) – A *RegisterStatusCallback* object. Pass in *None* to disable callbacks.

---

**Note:** Due to a risk of circular runaway leading to stack overflow, it is not allowed to call functions in the DLL from within the callback function. If a call is made to a function in the DLL the function will raise an exception.

---

## Examples

```
from ctypes import c_ubyte
from msl.equipment.resources import NKT

@NKT.RegisterStatusCallback
def register_callback(port, dev_id, reg_id, reg_status, reg_type,
    ↪length, address):
    # 'address' is an integer and represents the address of c_void_p
    ↪from the callback
    data = bytearray((c_ubyte * length).from_address(address)[:])
    print('The port is {!r}'.format(port))
    print('The device ID is {}'.format(dev_id))
    print('The register ID is {}'.format(reg_id))
    print('The register status is {!r}'.format(NKT.
    ↪RegisterStatusTypes(reg_status)))
    print('The register type is {!r}'.format(NKT.
    ↪RegisterDataTypes(reg_type)))
    print('The register data is {!r}'.format(data))

NKT.set_callback_register_status(register_callback)
```

`msl.equipment.resources.nkt.nktpdll.unknown_error(result)`

## msl.equipment.resources.omega package

Resources for equipment from OMEGA.

## Submodules

### msl.equipment.resources.omega.ithx module

OMEGA iTHX Series Temperature and Humidity Chart Recorder.

This class is compatible with the following model numbers:

- iTHX-W3
- iTHX-D3
- iTHX-SD
- iTHX-M
- iTHX-W
- iTHX-2

**class** `msl.equipment.resources.omega.ithx.iTHX(record)`

Bases: `ConnectionSocket`

OMEGA iTHX Series Temperature and Humidity Chart Recorder.

The *properties* for an iTHX connection supports the following key-value pairs in the *Connections Database*:

'nprobes': `int`, the number of probes the device has  
 'nbytes': `int`, the number of bytes to read from each probe

as well as those key-value pairs supported by the parent *ConnectionSocket* class.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**temperature**(*probe=1, celsius=True, nbytes=None*)

Read the temperature.

#### Parameters

- **probe** (`int`, optional) – The probe number to read the temperature of (for iTHX's that contain multiple probes).
- **celsius** (class:*bool*, optional) – `True` to return the temperature in celsius, `False` for fahrenheit.
- **nbytes** (class:*int*, optional) – The number of bytes to read. If `None` then read until the termination character sequence.

#### Returns

`float` or `tuple` of `float` – The temperature.

**humidity**(*probe=1, nbytes=None*)

Read the percent humidity.

#### Parameters

- **probe** (`int`, optional) – The probe number to read the humidity of (for iTHX's that contain multiple probes).
- **nbytes** (class:*int*, optional) – The number of bytes to read. If `None` then read until the termination character sequence.

#### Returns

`float` or `tuple` of `float` – The percent humidity.

**dewpoint**(*probe=1, celsius=True, nbytes=None*)

Read the dew point.

#### Parameters

- **probe** (`int`, optional) – The probe number to read the dew point of (for iTHX's that contain multiple probes).
- **celsius** (`bool`, optional) – `True` to return the dew point in celsius, `False` for fahrenheit.
- **nbytes** (class:*int*, optional) – The number of bytes to read. If `None` then read until the termination character sequence.

**Returns**

`float` or `tuple` of `float` – The dew point.

**temperature\_humidity**(*probe=1, celsius=True, nbytes=None*)

Read the temperature and the humidity.

**Parameters**

- **probe** (`int`, optional) – The probe number to read the temperature and humidity of (for iTHX's that contain multiple probes).
- **celsius** (`bool`, optional) – `True` to return the temperature in celsius, `False` for fahrenheit.
- **nbytes** (class:`int`, optional) – The number of bytes to read. If `None` then read until the termination character sequence. If specified, *nbytes* is the combined value to read both values.

**Returns**

- `float` – The temperature.
- `float` – The humidity.

**temperature\_humidity\_dewpoint**(*probe=1, celsius=True, nbytes=None*)

Read the temperature, the humidity and the dew point.

**Parameters**

- **probe** (`int`, optional) – The probe number to read the temperature, humidity and dew point (for iTHX's that contain multiple probes).
- **celsius** (`bool`, optional) – If `True` then return the temperature and dew point in celsius, `False` for fahrenheit.
- **nbytes** (`int`, optional) – The number of bytes to read. If `None` then read until the termination character sequence. If specified, *nbytes* is the combined value to read all three values.

**Returns**

- `float` – The temperature.
- `float` – The humidity.
- `float` – The dew point.

**reset**(*wait=True, password=None, port=2002, timeout=10*)

Power reset the iServer.

Some iServers accept the reset command to be sent via the TCP/UDP protocol and some require the reset command to be sent via the Telnet protocol.

**Parameters**

- **wait** (`bool`, optional) – Whether to wait for the connection to the iServer to be re-established before returning to the calling program. Re-booting an iServer takes about 10 to 15 seconds.
- **password** (`str`, optional) – The administrator's password of the iServer. If not specified then uses the default manufacturer's password. Only used if the iServer needs to be reset via the Telnet protocol.



- **port** (`int`, optional) – The port to use for the Telnet connection. Only used if the iServer needs to be reset via the Telnet protocol.
- **timeout** (`float`, optional) – The timeout value to use during the Telnet session. Only used if the iServer needs to be reset via the Telnet protocol.

**start\_logging**(*path*, *wait*=60, *nprobes*=None, *nbytes*=None, *celsius*=True, *msg\_format*=None, *db\_timeout*=10, *validator*=None)

Start logging the temperature, humidity and dew point to the specified path.

The information is logged to an [SQLite](#) database. To stop logging press CTRL+C.

#### Parameters

- **path** (`str`) – The path to the [SQLite](#) database. If you only specify a directory then a database with the default filename, `model_serial.sqlite3`, is created/opened in this directory.
- **wait** (`int`, optional) – The number of seconds to wait between each log event.
- **nprobes** (`int`, optional) – The number of probes that the iServer has (1 or 2). If not specified then gets the value defined in [properties](#). Default is 1.
- **nbytes** (`int`, optional) – The number of bytes to read from each probe (the probes are read sequentially). The value is passed to [temperature\\_humidity\\_dewpoint\(\)](#). If not specified then gets the value defined in [properties](#). Default is `None`.
- **celsius** (`bool`, optional) – `True` to return the temperature and dew point in celsius, `False` for fahrenheit.
- **msg\_format** (`str`, optional) – The format to use for the INFO logging messages each time data is read from an iServer. The format must use the `str.format()` syntax, `{}`. The positional arguments to `str.format()` are the values from the iServer, where the values are (*temperature*, *humidity*, *dewpoint*) for a 1-probe sensor and (*temperature1*, *humidity1*, *dewpoint1*, *temperature2*, *humidity2*, *dewpoint2*) for a 2-probe sensor. The keyword arguments to `str.format()` are the attributes of an [EquipmentRecord](#).

Examples:

- `T={0} H={1} D={2}`
- `{connection[address]} T={0:.1f} H={1:.1f} D={2:.1f}`
- `T1={0} T2={3} H1={1} H2={4} D1={2} D2={5}`
- `{alias} {serial} -> T={0}C H={1}% D={2}C`

- **db\_timeout** (`float`, optional) – The number of seconds the connection to the database should wait for the lock to go away until raising an exception.
- **validator** – A callback that is used to validate the data. The callback must accept two arguments (*data*, *itx*), where *data* is a [tuple](#) of the temperature, humidity and dewpoint values for each probe and *itx* is

the *iTHX* instance (i.e., *self*). The callback must return a value whose truthness decides whether to insert the data into the database. If the returned value evaluates to `True` then the data is inserted into the database.

**static data**(*path*, *start=None*, *end=None*, *as\_datetime=True*, *select='\*'*)

Fetch all the log records between two dates.

#### Parameters

- **path** (*str*) – The path to the `SQLite` database.
- **start** (*datetime* or *str*, optional) – Include all records that have a timestamp  $\geq$  *start*. If a *str* then in the ISO 8601 `yyyy-mm-dd` or `yyyy-mm-ddTHH:MM:SS` format.
- **end** (*datetime* or *str*, optional) – Include all records that have a timestamp  $\leq$  *end*. If a *str* then in the ISO 8601 `yyyy-mm-dd` or `yyyy-mm-ddTHH:MM:SS` format.
- **as\_datetime** (*bool*, optional) – Whether to fetch the timestamps in the database as `datetime` objects. If `False` then the timestamps will be of type *str* and this function will return much faster if requesting data over a large date range.
- **select** (*str* or *list* of *str*, optional) – The field name(s) in the database table to use for the SELECT SQL command (e.g., `'datetime'`, `'temperature'` or `['datetime', 'humidity']`).

#### Returns

*list* of *tuple* – A list of (*pid*, *datetime*, *temperature*, *humidity*, *dewpoint*, ...) log records, depending on the value of *select*.

`msl.equipment.resources.omega.ithx.convert_datetime(value)`

Convert a date and time to a `datetime` object.

#### Parameters

**value** (*bytes*) – The datetime value from an `SQLite` database.

#### Returns

`datetime.datetime` – The *value* as a datetime object.

## `msl.equipment.resources.optosigma` package

Resources for equipment from `OptoSigma`.

### Submodules

## `msl.equipment.resources.optosigma.shot702` module

Two-axis stage controller (SHOT-702) from `OptoSigma`.

**class** `msl.equipment.resources.optosigma.shot702.SHOT702(record)`

Bases: `ConnectionSerial`

Two-axis stage controller (SHOT-702) from `OptoSigma`.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**get\_input\_status()**

Get the I/O input connector status.

**Returns**

**status** (*int*) – Can either be 0 or 1 – see manual.

**get\_speed()**

Get the speed that each stage moves to a position.

**Returns**

*dict* –

The speed of each stage. The returned value has the form:

```
{
  'stage1' : (minimum, maximum, acceleration),
  'stage2' : (minimum, maximum, acceleration),
}
```

**get\_speed\_origin()**

Get the speed that each stage moves to the origin.

**Returns**

*dict* –

The speed of each stage. The returned value has the form:

```
{
  'stage1' : (minimum, maximum, acceleration),
  'stage2' : (minimum, maximum, acceleration),
}
```

**get\_steps()**

Get the number of steps for each stage.

**Returns**

- *int* – The number of steps for stage 1.
- *int* – The number of steps for stage 2.

**get\_travel\_per\_pulse()**

Get the travels per pulse for each stage.

**Returns**

- *float* – The travel per pulse for stage 1.
- *float* – The travel per pulse for stage 2.

**get\_version()**

Get the version number.

**Returns**

**str** – The version number.

**home(*stage*)**

Move the stage(s) to the home position.

**Parameters**

**stage** (**int** or **str**) – The stage(s) to home. Allowed values:

- 1 (home stage 1),
- 2 (home stage 2), or
- 'W' (home stages 1 and 2).

**Raises**

**OptoSigmaError** – If there was an error processing the command.

**is\_moving()**

Whether a stage is busy moving.

**Returns**

**bool** – Whether a stage is busy moving.

**move(*stage*, *direction*)**

Start moving the stage(s), at the minimum speed, in the specified direction.

**Parameters**

- **stage** (**int** or **str**) – The stage(s) to move. Allowed values:
  - 1 (start moving stage 1),
  - 2 (start moving stage 2), or
  - 'W' (start moving stages 1 and 2).
- **direction** (**str**) – The direction that the stage(s) should move. Allowed values are:
  - '+' or '-' (move a single stage in the specified direction)
  - '++' (move stage 1 in the + direction, stage 2 in the + direction)
  - '+-' (move stage 1 in the + direction, stage 2 in the - direction)
  - '-+' (move stage 1 in the - direction, stage 2 in the + direction)
  - '--' (move stage 1 in the - direction, stage 2 in the - direction)

**Raises**

**OptoSigmaError** – If there was an error processing the command.

**move\_absolute(*stage*, *\*position*)**

Move the stage(s) to the specified position.

Examples:

- move\_absolute(1, 1000)

- move stage 1 to position 1000 in the + direction
- `move_absolute(2, -5000)`
  - move stage 2 to position 5000 in the - direction
- `move_absolute('W', 1000, -5000)`
  - move stage 1 to position 1000 in the + direction, and
  - move stage 2 to position 5000 in the - direction

#### Parameters

- **stage** (`int` or `str`) – The stage(s) to move. Allowed values: 1, 2, 'W'.
- **position** (`int` or `tuple` of `int`) – The position the stage(s) should move to.

#### Raises

*OptoSigmaError* – If there was an error processing the command.

#### `move_relative(stage, *num_pulses)`

Move the stage(s) by a relative amount.

Examples:

- `move_relative(1, 1000)`
  - move stage 1 by 1000 pulses in the + direction
- `move_relative(2, -5000)`
  - move stage 2 by 5000 pulses in the - direction
- `move_relative('W', 1000, -5000)`
  - move stage 1 by 1000 pulses in the + direction, and
  - move stage 2 by 5000 pulses in the - direction

#### Parameters

- **stage** (`int` or `str`) – The stage(s) to move. Allowed values: 1, 2, 'W'.
- **num\_pulses** (`int` or `tuple` of `int`) – The number of pulses the stage(s) should move.

#### Raises

*OptoSigmaError* – If there was an error processing the command.

#### `set_mode(stage, mode)`

Set whether the stage(s) can be moved by hand or by the motor.

#### Parameters

- **stage** (`int` or `str`) – The stage(s) to set the mode of. Allowed values:
  - 1 (set the mode for stage 1),
  - 2 (set the mode for stage 2), or
  - 'W' (set the mode for stages 1 and 2).

- **mode** (*int*) – Whether the stage(s) can be moved by hand (0) or motor (1).

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**set\_origin**(*stage*)

Set the origin of the stage(s) to its current position.

**Parameters**

**stage** (*int* or *str*) – The stage(s) to set the home of. Allowed values:

- 1 (set the home for stage 1),
- 2 (set the home for stage 2), or
- 'W' (set the home for stages 1 and 2).

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**set\_output\_status**(*status*)

Set the I/O output status.

**Parameters**

**status** (*int*) – Can either be 0 or 1 – see manual.

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**set\_speed**(*stage, minimum, maximum, acceleration*)

Set the minimum, maximum and acceleration values when moving to a position.

Examples:

- `set_speed(1, 100, 1000, 50)`
  - set stage 1 to a minimum speed of 100 PPS, maximum speed of 1000 PPS and a 50 ms acceleration/deceleration time.
- `set_speed(2, 1000, 5000, 200)`
  - set stage 2 to a minimum speed of 1000 PPS, maximum speed of 5000 PPS and a 200 ms acceleration/deceleration time.
- `set_speed('W', [100,1000], [1000,5000], [50,200])`
  - set stage 1 to a minimum speed of 100 PPS, maximum speed of 1000 PPS and a 50 ms acceleration/deceleration time.
  - set stage 2 to a minimum speed of 1000 PPS, maximum speed of 5000 PPS and a 200 ms acceleration/deceleration time.

**Parameters**

- **stage** (*int* or *str*) – The stage(s) to set the setting for. Allowed values: 1, 2, 'W'.
- **minimum** (*int* or *list* of *int*) – The minimum speed (allowed range 1 - 500k).

- **maximum** (*int* or *list* of *int*) – The maximum speed (allowed range 1 - 500k).
- **acceleration** (*int* or *list* of *int*) – The acceleration and deceleration time in milliseconds (allowed range 1 - 1000).

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**set\_speed\_origin**(*stage, minimum, maximum, acceleration*)

Set the minimum, maximum and acceleration values when moving to the origin.

Examples:

- `set_speed_origin(1, 100, 1000, 50)`
  - set origin speed for stage 1 to a minimum speed of 100 PPS, maximum speed of 1000 PPS and a 50 ms acceleration/deceleration time.
- `set_speed_origin(2, 1000, 5000, 200)`
  - set origin speed for stage 2 to a minimum speed of 1000 PPS, maximum speed of 5000 PPS and a 200 ms acceleration/deceleration time.
- `set_speed_origin('W', [100,1000], [1000,5000], [50,200])`
  - set origin speed for stage 1 to a minimum speed of 100 PPS, maximum speed of 1000 PPS and a 50 ms acceleration/deceleration time.
  - set origin speed for stage 2 to a minimum speed of 1000 PPS, maximum speed of 5000 PPS and a 200 ms acceleration/deceleration time.

**Parameters**

- **stage** (*int* or *str*) – The stage(s) to set the setting for. Allowed values: 1, 2, 'W'.
- **minimum** (*int* or *list* of *int*) – The minimum origin speed (allowed range 1 - 500k).
- **maximum** (*int* or *list* of *int*) – The maximum origin speed (allowed range 1 - 500k).
- **acceleration** (*int* or *list* of *int*) – The origin acceleration and deceleration time in milliseconds (allowed range 1 - 1000).

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**set\_steps**(*stage, num\_steps*)

Set the number of steps that the stage motor will use.

See the manual for more details – the S command.

**Parameters**

- **stage** (*int*) – The stage to set the steps of (must be 1 or 2).
- **num\_steps** (*int*) – The number of steps that the motor should use (must be one of 1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 80, 100, 125, 200, 250).

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**status()**

Returns the current position and state of each stage.

**Returns**

- **pos1** (*int*) – The current position of stage 1.
- **pos2** (*int*) – The current position of stage 2.
- **state** (*str*) – The stopped state of the stage (one of 'L', 'M', 'W', 'K') – see the manual for more details.
- **is\_moving** (*bool*) – Whether a stage is busy moving.

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**stop()**

Immediately stop both stages from moving.

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**stop\_slowly(stage)**

Slowly bring the stage(s) to a stop.

**Parameters**

**stage** (*int* or *str*) – The stage(s) to slowly stop. Allowed values:

- 1 (slowly stop stage 1),
- 2 (slowly stop stage 2), or
- 'W' (slowly stop stages 1 and 2).

**Raises**

*OptoSigmaError* – If there was an error processing the command.

**wait(callback=None, sleep=0.05)**

Wait for the stages to finish moving.

This is a blocking call because it uses `time.sleep()`.

**Parameters**

- **callback** (*callable()*, optional) – A callable function. The function will receive 4 arguments – the returned values from `status()`
- **sleep** (*float*, optional) – The number of seconds to wait between calls to `callback`.



## msl.equipment.resources.optronic\_laboratories package

Resources for equipment from [Optronic Laboratories](#).

### Submodules

## msl.equipment.resources.optronic\_laboratories.ol756ocx\_32 module

Load the 32-bit OL756SDKActiveXCtrl library using [MSL-LoadLib](#).

```
class msl.equipment.resources.optronic_laboratories.ol756ocx_32.OL756(host, port,  
                                                                **kwargs)
```

Bases: [Server32](#)

Communicates with the 32-bit OL756SDKActiveXCtrl library.

**accumulate\_signals**(*meas\_type*)

Function needs to be called after a measurement was performed.

This essentially accumulates the data together until the user is ready to average out the data. This function is used in combination with [reset\\_averaging\(\)](#) and [do\\_averaging\(\)](#).

#### Parameters

**meas\_type** (*int*) – The measurement type wanted.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance
- 3 - Irradiance Calibration
- 4 - Radiance Calibration
- 5 - Transmittance Calibration

**connect\_to\_ol756**(*mode, com\_port=1*)

Desired mode to connect to OL756. If attempting to connect in RS232 or USB mode, and OL756 is not detected, then a dialog box will appear to prompt user to select either to retry, cancel or switch to DEMO.

#### Parameters

- **mode** (*int*) – Valid modes are:
  - -1: Disconnect. Call this before quitting the application.
  - 0: RS232
  - 1: USB
  - 2: DEMO mode
- **com\_port** (*int*, optional) – If connecting through RS232 then *port* is the COM port number to use.

#### Returns

*int* – The mode that was actually used for the connection.

**do\_averaging**(*meas\_type*, *num\_to\_average*)

Function divides the accumulated signal by the number of scans performed. It then sets the array containing the data with the averaged data. This function is used in combination with [reset\\_averaging\(\)](#) and [accumulate\\_signals\(\)](#).

**Parameters**

- **meas\_type** (*int*) – The measurement type wanted.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration
- **num\_to\_average** (*int*) – The number of scans to average.

**do\_calculations**(*meas\_type*)

Function needs to be called after each measurement to update the calculations.

**Parameters**

**meas\_type** (*int*) – The measurement type wanted.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance
- 3 - Irradiance Calibration
- 4 - Radiance Calibration
- 5 - Transmittance Calibration

**enable\_calibration\_file**(*meas\_type*, *enable*)

Enables or disables the use of a calibration file.

Use this option to generate calibrated results. To open a standard file used to create a calibration, use [enable\\_standard\\_file\(\)](#) instead.

The user should call [load\\_calibration\\_file\(\)](#) first to load the calibration file before enabling it.

**Parameters**

- **meas\_type** (*int*) – The measurement type wanted.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance
- **enable** (*bool*) – Whether to enable or disable the use of a calibration file.

**enable\_dark\_current**(*enable*)

Turn the dark current on or off.

Enable this feature if you want the dark current automatically acquired and subtracted before each measurement. If you wish to take a dark current manually, see the [get\\_dark\\_current\(\)](#) function.

The parameters for the dark current will need to be set using [set\\_dark\\_current\\_params\(\)](#).

**Parameters**

**enable** (*bool*) – Whether to turn the dark current on or off.

**enable\_pmt\_protection\_mode**(*enable*)

Turn the PMT protection routines on or off.

Enable this feature if you want the PMT to be shielded while traveling through high intensity spikes. This feature will make the scan slower since the wavelength and filter drive will move asynchronously.

The PMT is still protected by the hardware. This function prevents exposure of the PMT while traveling.

**Parameters**

**enable** (*bool*) – Whether to turn the PMT protection routines on or off.

**enable\_standard\_file**(*meas\_type*, *enable*)

Function enables standard files to be used.

To open a calibration file used to create a measurement, use [enable\\_calibration\\_file\(\)](#) instead.

The user should call [load\\_standard\\_file\(\)](#) first to load the standard file before enabling it.

**Parameters**

- **meas\_type** (*int*) – The calibration measurement type wanted.
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration
- **enable** (*bool*) – Whether to turn the application of the standard file on or off.

**export\_config\_file**(*file\_path*)

Exports the config file into a OL756 compatible configuration file.

Not all settings used will be applicable.

**Parameters**

**file\_path** (*str*) – A valid path to save the file at.

**export\_registry**()

Save data out to the Windows registry.

Make sure that a read was done at some point using `import_registry()`. Does not create a configuration file that can be loaded into another computer. For that particular function, call `export_config_file()`.

**get\_adaptive\_int\_time\_index**(*gain\_index*)

Get the adaptive integration time index.

**Parameters**

**gain\_index** (*int*) – The index of the gain to use to get the integration time.

- 0 - 1.0E-5
- 1 - 1.0E-6
- 2 - 1.0E-7
- 3 - 1.0E-8
- 4 - 1.0E-9
- 5 - 1.0E-10
- 6 - 1.0E-11

**Returns**

*int* – The adaptive integration time index.

**get\_cri**(*meas\_type*, *index*)

Get the color-rendering information.

The user should call `do_calculations()` at least once before calling this function.

**Parameters**

- **meas\_type** (*int*) – The measurement type wanted.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration
- **index** (*int*) – The color-rendering index.
  - 0 - General CRI
  - 1 - Light Greyish Red (CRI#1)
  - 2 - Dark Greyish Yellow (CRI#2)
  - 3 - Strong Yellow Green (CRI#3)
  - 4 - Moderate Yellowish Green (CRI#4)
  - 5 - Light Bluish Green (CRI#5)
  - 6 - Light Blue (CRI#6)
  - 7 - Light Violet (CRI#7)

- 8 - Light Reddish Purple (CRI#8)
- 9 - Strong Red (CRI#9)
- 10 - Strong Yellow (CRI#10)
- 11 - Strong Green (CRI#11)
- 12 - Strong Blue (CRI#12)
- 13 - Light Yellowish Pink (CRI#13)
- 14 - Moderate Olive Green (CRI#14)

**Returns**

`float` – The color-rendering information.

**get\_cal\_array()**

This method allows user to get the spectral data of a calibration after it is made. The data allows the user to take the data and create their own data files.

**Returns**

- `int` – A pointer to an array of signals.
- `int` – The number of points acquired.

**get\_cal\_file\_enabled(*meas\_type*)**

Checks to see if the calibration file is enabled.

The user should call `load_calibration_file()` first to load the calibration file before enabling it.

**Parameters**

**meas\_type** (`int`) – The measurement type wanted.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance

**Returns**

`bool` – Whether the calibration file is enabled.

**get\_calculated\_data(*meas\_type*, *index*)**

Gets data calculated from the intensities.

The user should call `do_calculations()` at least once before calling this function.

**Parameters**

- **meas\_type** (`int`) – The measurement type wanted.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration

- **index** (*int*) – The index to retrieve data of.
  - 0 - Color Temperature
  - 1 - Dominant Wavelength
  - 2 - LED Half Bandwidth
  - 3 - Left Half Bandwidth
  - 4 - Right Half Bandwidth
  - 5 - Peak Spectral Value
  - 6 - LEDPeakWavelength
  - 7 - Radiometric Value
  - 8 - Purity
  - 9 - Center Wavelength
  - 10 - Photometric Value

**Returns**

*float* – Pointer to a double to hold the data.

**get\_calibration\_file**(*meas\_type*)

Get a calibration file that is loaded.

**Parameters**

**meas\_type** (*int*) – The measurement type wanted.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance

**Returns**

*str* – String containing the name and path of the calibration file that is loaded for a particular measurement type.

**get\_chromaticity\_data**(*meas\_type*, *index*)

Get the calculated chromaticity values requested.

Must have called *do\_calculations()* at least once.

**Parameters**

- **meas\_type** (*int*) – The measurement type wanted.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration

- **index** (*int*) – The chromaticity index value [0..70]. See the SDK manual for more details.

**Returns**

*float* – Pointer to a double to hold the data.

**get\_dark\_current**(*use\_compensation*)

Takes a manual dark current.

User will have to subtract from data array by retrieving this array via a *get\_cal\_array()* or *get\_signal\_array()*. This is a special function and most users will want to use *enable\_dark\_current()* instead because it automatically does the subtraction.

Function if called externally by user program will not have result saved out to data file. If the *enable\_dark\_current()* was enabled, then this function need should not be called.

**Parameters**

**use\_compensation** (*int*) – Adjusts dark current for more dynamic ranging using reverse current.

**Returns**

*float* – The dark current.

**get\_dark\_current\_enable**()

Returns whether the dark-current mode is enabled.

**Returns**

*bool* – Whether the dark-current mode is enabled or disabled.

**get\_dark\_current\_mode**()

Returns whether the dark current is taken at a wavelength or in shutter mode.

**Returns**

*int* – The dark-current mode

- 0 - Dark current in wavelength mode (taken at a particular wavelength designated by the user).
- 1 - Dark current in shutter mode

**get\_dark\_current\_wavelength**()

Get the dark current wavelength.

**Returns**

*float* – Wavelength that the dark current will be taken at.

**get\_ending\_wavelength**()

Get the ending wavelength of the scan range.

**Returns**

*float* – The ending wavelength, in nanometers, of the scan range.

**get\_gain\_index**()

Get the index of the gain that will be applied when the parameters are to be sent down.

Applies to both quick scan and point to point scans.

**Returns**

*int* – The gain index.

- 0 - 1.0E-5
- 1 - 1.0E-6
- 2 - 1.0E-7
- 3 - 1.0E-8
- 4 - 1.0E-9
- 5 - 1.0E-10 (Point to Point mode only)
- 6 - 1.0E-11 (Point to Point mode only)
- 7 - Auto Gain Ranging (Point to Point mode only)

#### **get\_increment()**

Get the wavelength increment that is used for a scan.

##### **Returns**

**float** – The wavelength increment, in nanometers.

#### **get\_increment\_index()**

Get the index of the wavelength increment that is used for a scan.

Applies to both quick scan and point to point scans.

##### **Returns**

**int** – Index of the wavelength increment of a scan.

- 0 - 0.025 nm
- 1 - 0.05 nm
- 2 - 0.1 nm
- 3 - 0.2 nm
- 4 - 0.5 nm
- 5 - 1.0 nm
- 6 - 2.0 nm
- 7 - 5.0 nm
- 8 - 10.0 nm

#### **get\_integration\_time\_index(scan\_mode)**

Get the index into the integration time array.

Applies to both quick scan and point to point scans. In quick scan, the speed will vary based on the scan range and increments.

##### **Parameters**

**scan\_mode** (**int**) – The scan mode to use to get the index of.

##### **Returns**

**int** – Point to Point mode

- 0 - 1.000 sec
- 1 - 0.500 sec



- 2 - 0.200 sec
- 3 - 0.100 sec
- 4 - 0.050 sec
- 5 - 0.020 sec
- 6 - 0.010 sec
- 7 - 0.005 sec
- 8 - 0.002 sec
- 9 - 0.001 sec
- 10 - Adaptive (Point To Point mode only)

Quick Scan mode

- 0 - slowest
- 10 - fastest

#### **get\_ocx\_version()**

Get the version of the OL756 SDK ActiveX control.

##### **Returns**

`str` – The software version.

#### **get\_pmt\_flux\_overload()**

Get the voltage of the photomultiplier tube flux overload.

##### **Returns**

`float` – Voltage that the PMT will determine to be at the overload point.

#### **get\_pmt\_voltage()**

Returns the voltage that will sent or has been sent down to the PMT.

##### **Returns**

`float` – Voltage value, in volts, of the photomultiplier tube.

#### **get\_quick\_scan\_rate()**

Returns the rate at the quick scan index.

##### **Returns**

`float` – Rate of the quick scan at the current index in nm/s.

#### **get\_quick\_scan\_rate\_index()**

Returns the index of the quick scan rate.

##### **Returns**

`int` – Index of the quick scan rate.

#### **get\_scan\_mode()**

Get the mode the scan will be done in.

##### **Returns**

`int` – The scan mode

- 0 - Point to Point mode

- 1 - Quick Scan mode

#### **get\_settling\_time()**

Get the settling time.

Settling time is time where the wavelength drive pauses once it reaches its target wavelength.

##### **Returns**

**float** – Settling time, in seconds, to be sent down or has already been sent to the system.

#### **get\_signal\_array()**

Get the spectral data of a measurement after it is made.

##### **Returns**

**tuple** – The spectral data.

#### **get\_standard\_file(*meas\_type*)**

Retrieves the name of standard file.

##### **Parameters**

**meas\_type** (**int**) – The measurement type wanted.

- 3 - Irradiance calibration
- 4 - Radiance calibration
- 5 - Transmittance calibration

##### **Returns**

**str** – String containing the name and path of the standard file that is loaded for a particular calibration type.

#### **get\_start\_wavelength()**

Get the starting wavelength of a scan.

Applies to both quick scan and point to point scans.

##### **Returns**

**float** – The wavelength, in nanometers, that the scan will start from.

#### **get\_std\_file\_enabled(*meas\_type*)**

Checks to see if the standard file is enabled.

The user should call [\*load\\_standard\\_file\(\)\*](#) first to load the standard file before enabling it.

##### **Parameters**

**meas\_type** (**int**) – The calibration type wanted.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance

##### **Returns**

**bool** – Whether a standard file is enabled.

**import\_config\_file**(*path*)

The file is a standard OL756 configuration file.

Not all settings used will be applicable. Measurement type is not used because in the SDK, the `take_point_to_point_measurement()` function has as an input the measurement type. The user should select the type and not have it based on the configuration file.

**Parameters**

**path** (*str*) – A valid path to load the file at.

**import\_registry**()

Loads data from the registry.

Loads default if no registry exists. To import the configuration from another computer, use `import_config_file()` instead.

Not all settings used will be applicable. Measurement type is not used because in the SDK, the `take_point_to_point_measurement()` function has as an input the measurement type. The user should select the type and not have it based on the configuration file.

**load\_calibration\_file**(*path, meas\_type*)

Load a calibration file.

**Parameters**

- **path** (*str*) – The path of a calibration file.
- **meas\_type** (*int*) – The measurement type.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance

**load\_standard\_file**(*path, meas\_type*)

Load a standard file.

**Parameters**

- **path** (*str*) – The path of a standard file.
- **meas\_type** (*int*) – The measurement type.
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration

**manual\_filter\_drive\_connect**(*connect*)

Used to connect or disconnect the filter drive.

Disconnecting essentially acquires scans without the filter.

**Parameters**

**connect** (*bool*) – Connect or disconnect the filter drive. Reconnecting will home the wavelength and filter drive.

### **manual\_get\_gain()**

The index of the gain that will be applied when the parameters are to be sent down.

#### **Returns**

`int` – The gain index.

- 0 - 1.0E-5
- 1 - 1.0E-6
- 2 - 1.0E-7
- 3 - 1.0E-8
- 4 - 1.0E-9
- 5 - 1.0E-10 (Point to Point mode only)
- 6 - 1.0E-11 (Point to Point mode only)
- 7 - Auto Gain Ranging (Point to Point mode only)

### **manual\_get\_integration\_time()**

Returns the integration time set in the system.

Only applies to the integration time used for Point to Point scans.

#### **Returns**

`float` – The integration time in seconds.

### **manual\_get\_pmt\_overload()**

Returns the PMT overload voltage set in the system.

#### **Returns**

`float` – Overload voltage, in volts, of the photomultiplier tube.

### **manual\_get\_pmt\_voltage()**

Returns the PMT high voltage set in the system.

#### **Returns**

`float` – Voltage, in volts, of the photomultiplier tube.

### **manual\_get\_settling\_time()**

Returns the settling time of the instrument.

#### **Returns**

`float` – Settling time of the system in seconds.

### **manual\_get\_signal()**

Returns the signal at the current position of the wavelength drive.

#### **Returns**

`float` – The signal, in amperes.

### **manual\_home\_ol756()**

Homes the wavelength and filter drive.

Will reconnect the filter drive if it was disconnected

**manual\_move\_to\_wavelength**(*wavelength*)

Moves the wavelength drive to a particular location.

**Parameters**

**wavelength** (*float*) – The wavelength to move the wavelength drive to.

**manual\_set\_gain**(*gain\_index, mode*)

Set the gain.

**Parameters**

- **gain\_index** (*int*) – The gain index.
  - 0 - 1.0E-5
  - 1 - 1.0E-6
  - 2 - 1.0E-7
  - 3 - 1.0E-8
  - 4 - 1.0E-9
  - 5 - 1.0E-10 (Point to Point mode only)
  - 6 - 1.0E-11 (Point to Point mode only)
  - 7 - Auto Gain Ranging (Point to Point mode only)
- **mode** (*int*) – The scan mode
  - 0 - point to point
  - 1 - quick scan

**manual\_set\_integration\_time**(*time*)

Sets the integration time set in the system.

Only applies to the integration time used for Point to Point scans.

**Parameters**

**time** (*float*) – The integration time in seconds.

**manual\_set\_pmt\_overload**(*overload*)

Sets the PMT overload voltage set in the system.

**Parameters**

**overload** (*float*) – Overload voltage, in volts, of the photomultiplier tube in Volts.

**manual\_set\_pmt\_voltage**(*voltage*)

Sets the PMT high voltage set in the system.

**Parameters**

**voltage** (*float*) – Voltage, in volts, of the photomultiplier tube.

**manual\_set\_settling\_time**(*time*)

Sets the settling time of the instrument.

**Parameters**

**time** (*float*) – Settling time of the system.

**move\_to\_wavelength**(*wavelength*)

Moves the wavelength drive to a particular location.

**Parameters**

**wavelength** (*float*) – The wavelength, in nanometers, to move the wavelength drive to.

**read\_ol756\_flash\_settings**()

Reads the saved settings from the flash memory.

Reads the settings such as the grating alignment factor, filter skew and wavelength skew. Loads these values into the ActiveX control memory.

**reset\_averaging**(*meas\_type*)

Resets the accumulated signal array for the specified measurement type.

This function is used in combination with *do\_averaging()* and *accumulate\_signals()*.

**Parameters**

**meas\_type** (*int*) – The measurement type.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance
- 3 - Irradiance Calibration
- 4 - Radiance Calibration
- 5 - Transmittance Calibration

**save\_calibration\_file**(*meas\_type*, *path*)

Create a OL756-compatible calibration file.

**Parameters**

- **meas\_type** (*int*) – The measurement type.
  - 3 - Irradiance Calibration
  - 4 - Radiance Calibration
  - 5 - Transmittance Calibration
- **path** (*str*) – The path to save the calibration file to.

**save\_measurement\_data**(*meas\_type*, *path*)

Save the measurement data to a OL756-compatible data file.

**Parameters**

- **meas\_type** (*int*) – The measurement type.
  - 0 - Irradiance
  - 1 - Radiance
  - 2 - Transmittance
- **path** (*str*) – The path to save the data to.

**send\_down\_parameters**(*scan\_mode*)

Sends down the parameters to the system.

This needs to be called whenever parameters dealing with the PMT or integration time and gain has changed. Needs to be called once before doing any measurements or other signal acquisition including dark current.

The following methods affect the parameters: `set_pmt_flux_overload_voltage()`  
`set_gain()`            `set_integration_time()`            `set_pmt_hi_voltage()`  
`set_settling_time()` `set_scan_range()` `set_adaptive_integration_time()`

**Parameters**

**scan\_mode** (*int*) – The scan mode.

- 0 - Point to point
- 1 - Quick scan

**set\_adaptive\_integration\_time**(*gain\_index*, *speed\_index*)

Sets the scan speed of the scan at a particular gain range.

Adaptive integration time is used solely for point to point scans in auto-gain ranging.

**Parameters**

- **gain\_index** (*int*) – The index of the gain to use to set the integration time.
  - 0 - 1.0E-5
  - 1 - 1.0E-6
  - 2 - 1.0E-7
  - 3 - 1.0E-8
  - 4 - 1.0E-9
  - 5 - 1.0E-10
  - 6 - 1.0E-11
- **speed\_index** (*int*) – The scan speed index [0..12] – 0=Slowest, 12=Fastest.

**set\_averaging\_number\_of\_scan**(*num\_avg\_scans*)

Set the number of scans to average.

**Parameters**

**num\_avg\_scans** (*int*) – The number of scans to average.

**set\_dark\_current\_params**(*mode*, *wavelength*)

Sets the mode and the wavelength to use for a dark-current measurement.

**Parameters**

- **mode** (*int*) – The mode to use to acquire a dark-current measurement
  - 0 - wavelength
  - 1 - shutter

- **wavelength** (*float*) – The wavelength, in nanometers, to use for a dark-current measurement.

**set\_gain**(*scan\_mode*, *gain\_index*)

Sets the index of the gain that will be applied when the parameters are to be sent down.

Applies to both quick scan and point to point scans.

**Parameters**

- **scan\_mode** (*int*) – The scan mode
  - 0 - Point to Point
  - 1 - Quick Scan
- **gain\_index** (*int*) – The gain index
  - 0 - 1.0E-5
  - 1 - 1.0E-6
  - 2 - 1.0E-7
  - 3 - 1.0E-8
  - 4 - 1.0E-9
  - 5 - 1.0E-10 (available only in Point to Point mode)
  - 6 - 1.0E-11 (available only in Point to Point mode)
  - 7 - Auto Gain Ranging (available only in Point to Point mode)

**set\_integration\_time**(*scan\_mode*, *scan\_speed*)

Sets the index of the scan speed used.

Applies to both quick scan and point to point scans. In quick scan, the speed will vary based on the scan range and increments.

**Parameters**

- **scan\_mode** (*int*) – The scan mode
  - 0 - Point to Point
  - 1 - Quick Scan
- **scan\_speed** (*int*) – Index to the integration time array
  - Point to Point mode
    - 0 - 1.000 sec
    - 1 - 0.500 sec
    - 2 - 0.200 sec
    - 3 - 0.100 sec
    - 4 - 0.050 sec
    - 5 - 0.020 sec
    - 6 - 0.010 sec



- 7 - 0.005 sec
- 8 - 0.002 sec
- 9 - 0.001 sec
- 10 - Adaptive (Point To Point mode only)
- 11 - User defined (Point To Point mode only)

Quick Scan mode

- 0 - slowest
- 10 - fastest

**set\_pmt\_flux\_overload\_voltage**(*overload\_voltage*)

Sets the value to use for the photomultiplier tube flux overload.

**Parameters**

**overload\_voltage** (*float*) – Voltage that the PMT will determine to be at the overload point. Software only, because PMT has built-in protection also.

**set\_pmt\_hi\_voltage**(*hi\_voltage*)

Sets the value to be determined to be a flux overload by the software.

**Parameters**

**hi\_voltage** (*float*) – Voltage, in volts, that the PMT will determine to be overload point.

**set\_reference\_white\_point**(*white, user\_def\_x, user\_def\_y*)

Sets the value of the reference illuminant.

**Parameters**

- **white** (*int*) – The reference white point
  - 0 - Incandescent(A)
  - 1 - Direct Sunlight(B)
  - 2 - Indirect Sunlight(C)
  - 3 - Natural Daylight(D65)
  - 4 - Normalized Reference(E)
  - 5 - User Defined
- **user\_def\_x** (*float*) – User defined x on CIE chart.
- **user\_def\_y** (*float*) – User defined y on CIE chart.

**set\_scan\_range**(*start, end, inc\_index*)

Sets the wavelength scan range.

**Parameters**

- **start** (*float*) – Starting wavelength, in nanometers.
- **end** (*float*) – Ending wavelength, in nanometers.
- **inc\_index** (*int*) – Increment index, in nanometers.

**set\_settling\_time(*time*)**

Set the settling time.

Settling time is the time that the wavelength drive pauses once it reaches its target wavelength.

**Parameters**

**time** (*float*) – Settling Time in seconds to be sent down or has already been sent to the system.

**set\_tab\_delimited\_mode(*enable*)**

Purpose of function is to set what mode to write the data files as.

Setting the tab delimited to true will write the data in a tab delimited format, else a false will write in a comma delimited format. Tab delimited files will not be compatible with some versions of the software. If you want data files to be compatible with v1.32 software and below, leave the mode to *False*.

**Parameters**

**enable** (*bool*) – Whether to use the new file format using TABs as a delimited or the old file format compatible with v1.32 and below.

**set\_user\_defined\_integration\_time(*time*)**

Sets the user defined integration time to be used only in point to point scans and only if the user sets the integration time mode.

**Parameters**

**time** (*float*) – Integration time in seconds.

**stop\_measurement()**

Stops a measurement.

Applies only to Point to Point measurements. Quick scans are done so quickly that there is no need to stop a measurement once it starts.

**take\_point\_to\_point\_calibration(*meas\_type*)**

Takes a calibration in point to point mode.

Need to have called *send\_down\_parameters()* at least once before calling any of the measurement functions or data acquisition functions.

**Parameters**

**meas\_type** (*int*) – The measurement type.

- 3 - Irradiance Calibration
- 4 - Radiance Calibration
- 5 - Transmittance Calibration

**take\_point\_to\_point\_measurement(*meas\_type*)**

Takes a measurement in point to point mode.

Need to have called *send\_down\_parameters()* at least once before calling any of the measurement functions or data acquisition functions.

**Parameters**

**meas\_type** (*int*) – The measurement type.

- 0 - Irradiance

- 1 - Radiance
- 2 - Transmittance

**take\_quick\_scan\_calibration**(*meas\_type*)

Takes a calibration in quick scan mode.

Need to have called [send\\_down\\_parameters\(\)](#) at least once before calling any of the measurement functions or data acquisition functions.

**Parameters**

**meas\_type** (*int*) – The measurement type.

- 3 - Irradiance Calibration
- 4 - Radiance Calibration
- 5 - Transmittance Calibration

**take\_quick\_scan\_measurement**(*meas\_type*)

Takes a measurement in quick scan mode.

Need to have called [send\\_down\\_parameters\(\)](#) at least once before calling any of the measurement functions or data acquisition functions.

**Parameters**

**meas\_type** (*int*) – The measurement type.

- 0 - Irradiance
- 1 - Radiance
- 2 - Transmittance

## **msl.equipment.resources.optronic\_laboratories.ol756ocx\_64 module**

Load the 32-bit OL756SDKActiveXCtrl library using [MSL-LoadLib](#).

**class** `msl.equipment.resources.optronic_laboratories.ol756ocx_64.OL756`(*record=None*)

Bases: [Connection](#)

A wrapper around the OL756SDKActiveXCtrl library.

**Attention:** See the [ol756ocx\\_32.OL756](#) class for all available methods.

This class can be used with either a 32- or 64-bit Python interpreter to call the 32-bit functions in the OL756SDKActiveXCtrl library.

The [properties](#) for an OL756 connection supports the following key-value pairs in the [Connections Database](#):

```
'mode': int, connection mode (0=RS232, 1=USB) [default: 1]
'com_port': int, the COM port number (RS232 mode only) [default: 1]
```

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**disconnect()**

Disconnect from the OL756 spectroradiometer.

**msl.equipment.resources.optronic\_laboratories.ol\_current\_source module**

Communicate with a DC current source from Optronic Laboratories.

**class** msl.equipment.resources.optronic\_laboratories.ol\_current\_source.OLCurrentSource(*record*)

Bases: *ConnectionSerial*

Communicate with a DC current source from Optronic Laboratories.

**Attention:** The COM interface must be selected (using the buttons on the front panel) to be RS-232 after the Current Source is initially powered on. Even if this is the default power-on interface, it must be re-selected before communication will work. This is required for models with firmware revision 5.8, other firmware versions may be different.

The *properties* for the connection supports the following key-value pairs in the *Connections Database*:

'address': *int*, the internal address of the device [default: 1]  
'delay': *float*, the number of seconds to wait between send/receive.  
→ commands [default: 0.05]

as well as those key-value pairs supported by the parent *ConnectionSerial* class.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**get\_current()**

Get the output current.

**Returns**

*float* – The value of the output current.

**get\_setup(lamp, typ)**

Get the lamp setup.

**Parameters**

- **lamp** (*int*) – The lamp number, between 0 and 9.
- **typ** (*int*) – The type of data to read. Must be one of 40, 50, 60, 70, 80, 90, 95.
  - 40: Lamp Hours
  - 50: Recalibration interval (hours)

- 60: Target units (A, V or W)
- 70: Target value
- 80: Current limit
- 90: Lamp description text
- 95: Wattage (L or H)

**Returns**

`str` or `float` – The value of the data type that was requested.

**get\_voltage()**

Get the output voltage.

**Returns**

`float` – The value of the output voltage.

**get\_wattage()**

Get the output wattage.

**Returns**

`float` – The value of the output wattage.

**reset()**

Reset the communication buffers.

**select\_lamp(*number*)**

Select a lamp.

**Parameters**

**number** (`int`) – The lamp number to select, between 0 and 9.

**set\_current(*amps*)**

Set the target output current.

**Parameters**

**amps** (`float`) – The target current, in Amps. If the value is above the target current limit for the presently selected lamp setup or if the value is less than the minimum supported current, the target current will not change.

**Returns**

`float` – The present value of the output current.

**set\_setup(*lamp, typ, value*)**

Set the lamp setup.

**Parameters**

- **lamp** (`int`) – The lamp number, between 0 and 9.
- **typ** (`int`) – The type of data to write. Must be one of 40, 50, 60, 70, 80, 90, 95.
  - 40: Lamp Hours
  - 50: Recalibration interval (hours)
  - 60: Target units (A, V or W)
  - 70: Target value

- 80: Current limit
- 90: Lamp description text
- 95: Wattage (L or H)
- **value** (`str` or `float`) – The value to write.

**set\_voltage**(*volts*)

Set the target output voltage.

**Parameters**

**volts** (`float`) – The target voltage, in Volts. If the value is above the target voltage limit for the presently selected lamp setup or if the value is less than the minimum supported voltage, the target voltage will not change.

**Returns**

`float` – The present value of the output voltage.

**set\_wattage**(*watts*)

Set the target output wattage.

**Parameters**

**watts** (`float`) – The target wattage, in Volts. If the value is above the target wattage limit for the presently selected lamp setup or if the value is less than the minimum supported wattage, the target wattage will not change.

**Returns**

`float` – The present value of the output wattage.

**state**()

Returns whether the output is on or off.

**Returns**

`bool` – `True` if the output is on, `False` if the output is off.

**property system\_status\_byte**

The system status byte that is returned in every reply.

It is constructed as follows:

- bit 7: Busy flag (the device is performing a function)
- bit 6: Reserved
- bit 5: Reserved
- bit 4: Lamp status (0=off, 1=on)
- bit 3: Reserved
- bit 2: Reserved
- bit 1: Seeking current (1=current is ramping)
- bit 0: Reserved

**Type**

`int`

**target\_info()**

Get the target information of the currently-selected lamp.

**Returns**

`dict` – The lamp number, target value and target unit. The key-value pairs are:

```
{'lamp': int, 'value': float, 'unit': str}
```

**turn\_off()**

Turn the output off.

**turn\_on()**

Turn the output on.

**zero\_voltage\_monitor()**

Zero the voltage monitor.

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

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

**Subpackages****msl.equipment.resources.picotech.errors module**

Exceptions and error codes defined in the Pico Technology SDK.

```
class msl.equipment.resources.picotech.errors.PS2000Error(value, names=None, *,
                                                         module=None,
                                                         qualname=None,
                                                         type=None, start=1,
                                                         boundary=None)
```

Bases: `IntEnum`

**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.errors.PS3000Error(value, names=None, *,
                                                         module=None,
                                                         qualname=None,
                                                         type=None, start=1,
                                                         boundary=None)
```

Bases: `IntEnum`

**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

### 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.callbacks.BlockReady`

All BlockReady callbacks have the same function signature, so create a generic BlockReady callback.

`msl.equipment.resources.picotech.picoscope.callbacks.ps2000aBlockReady`

ps2000aBlockReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps3000aBlockReady`

ps3000aBlockReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps4000BlockReady`

ps4000BlockReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps4000aBlockReady`

ps4000aBlockReady callback



`msl.equipment.resources.picotech.picoscope.callbacks.ps5000BlockReady`  
ps5000BlockReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps5000aBlockReady`  
ps5000aBlockReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps6000BlockReady`  
ps6000BlockReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps2000aStreamingReady`  
ps2000aStreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps3000aStreamingReady`  
ps3000aStreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps4000StreamingReady`  
ps4000StreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps4000aStreamingReady`  
ps4000aStreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps5000StreamingReady`  
ps5000StreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps5000aStreamingReady`  
ps5000aStreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps6000StreamingReady`  
ps6000StreamingReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps2000aDataReady`  
ps2000aDataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps3000aDataReady`  
ps3000aDataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps4000DataReady`  
ps4000DataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps4000aDataReady`  
ps4000aDataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps5000DataReady`  
ps5000DataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps5000aDataReady`  
ps5000aDataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.ps6000DataReady`  
ps6000DataReady callback

`msl.equipment.resources.picotech.picoscope.callbacks.PS3000_CALLBACK_FUNC`  
PS3000\_CALLBACK\_FUNC callback

`msl.equipment.resources.picotech.picoscope.callbacks.GetOverviewBuffersMaxMin`  
GetOverviewBuffersMaxMin callback

**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_value)
```

Bases: `object`

Contains the information about a PicoScope channel.

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

**Parameters**

- **channel** (`enum.IntEnum`) – The PSX000xChannel enum.
- **enabled** (`bool`) – Whether the channel is enabled.
- **coupling** (`enum.IntEnum`) – The PSX000xCoupling enum, 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 PSX000xBandwidthLimiter enum.
- **max\_adu\_value** (`int`) – The maximum analog-to-digital unit.

**property channel**

The PSX000xChannel enum.

**Type**

`enum.IntEnum`

**property enabled**

Whether the channel is enabled.

**Type**

`bool`

**property coupling**

The PSX000xCoupling enum, e.g. AC or DC.

**Type**

`enum.IntEnum`

**property voltage\_range**

The voltage range, in Volts.

**Type**

`float`

**property voltage\_offset**

The voltage offset, in Volts.

**Type**

`float`

**property bandwidth**

The PSX000xBandwidthLimiter enum.

**Type**

`enum.IntEnum` or `None`

**property volts\_per\_adu**

The conversion factor to convert ADU to volts

**Type**

`float`

**property raw**

The raw data, in ADU

**Type**

`numpy.ndarray`

**property buffer**

The raw data, in ADU

**Type**

`numpy.ndarray`

**property volts**

The data, in volts

**Type**

`numpy.ndarray`

**property num\_samples**

The size of the data array.

**Type**

`int`

**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(value,  
                                                                    names=None,  
                                                                    *, mod-  
                                                                    ule=None,  
                                                                    qual-  
                                                                    name=None,  
                                                                    type=None,  
                                                                    start=1,  
                                                                    bound-  
                                                                    ary=None)
```

Bases: [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(value,  
                                                                    names=None,  
                                                                    *, mod-  
                                                                    ule=None,  
                                                                    qual-  
                                                                    name=None,  
                                                                    type=None,  
                                                                    start=1,  
                                                                    bound-  
                                                                    ary=None)
```

Bases: [IntEnum](#)

**A** = 0

**B** = 1

**C = 2**

**D = 3**

**EXT = 4**

**MAX\_CHANNELS = 4**

**NONE = 5**

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000Range(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: [IntEnum](#)

**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(value,
                                                                            names=None,
                                                                            *, mod-
                                                                            ule=None,
                                                                            qual-
                                                                            name=None,
                                                                            type=None,
                                                                            start=1,
                                                                            bound-
                                                                            ary=None)
```

Bases: `IntEnum`

**FS** = 0

**PS** = 1

**NS** = 2

**US** = 3

**MS** = 4

**S** = 5

**MAX** = 6

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000Info(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                                    names=None,
                                                                                    *,
                                                                                    mod-
                                                                                    ule=None,
                                                                                    qual-
                                                                                    name=None,
                                                                                    type=None,
                                                                                    start=1,
                                                                                    bound-
                                                                                    ary=None)
```

Bases: `IntEnum`

**RISING** = 0

**FALLING** = 1

**MAX** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS20000OpenProgress(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              boundary=None)
```

Bases: `IntEnum`

**FAIL** = -1

**PENDING** = 0

**COMPLETE** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS20000EtsMode(value,
                                                                           names=None,
                                                                           *, module=
                                                                           None, qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=
                                                                           None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS20000ButtonState(value,
                                                                              names=None,
                                                                              *, module=
                                                                              None, qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              boundary=
                                                                              None)
```

Bases: `IntEnum`

`NO_PRESS = 0`

`SHORT_PRESS = 1`

`LONG_PRESS = 2`

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000SweepType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`UP = 0`

`DOWN = 1`

`UPDOWN = 2`

`DOWNUP = 3`

`MAX = 4`

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000WaveType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`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(value,
names=None,
*,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Bases: `IntEnum`

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(value,
names=None,
*,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Bases: `IntEnum`

LEVEL = 0

WINDOW = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000TriggerState(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

DONT\_CARE = 0

TRUE = 1

FALSE = 2

MAX = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000PulseWidthType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

NONE = 0

LESS\_THAN = 1

GREATER\_THAN = 2

IN\_RANGE = 3

OUT\_OF\_RANGE = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex(value,
                                                                                   names=None,
                                                                                   *,
                                                                                   mod-
                                                                                   ule=None,
                                                                                   qual-
                                                                                   name=None,
                                                                                   type=None,
                                                                                   start=1,
                                                                                   bound-
                                                                                   ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                                   names=None,
                                                                                   *, mod-
                                                                                   ule=None,
                                                                                   qual-
                                                                                   name=None,
                                                                                   type=None,
                                                                                   start=1,
                                                                                   bound-
                                                                                   ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**OR** = 1

**AND** = 2

**THEN** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000DigitalPort(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**PORT0** = 128

**PORT1** = 129

**PORT2** = 130

**PORT3** = 131

**MAX** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalChannel(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

```
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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**AC = 0**

**DC = 1**

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelInfo(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**RANGES = 0**

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AEtsMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000ATimeUnits(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**FS** = 0

**PS** = 1

**NS** = 2

**US** = 3

**MS** = 4

**S** = 5

**MAX** = 6

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000ASweepType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**UP** = 0

**DOWN** = 1

**UPDOWN** = 2

**DOWNUP** = 3

**MAX** = 4



```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
    Bases: IntEnum
```

```
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(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
    Bases: IntEnum
```

```
OFF = 0
```

```
WHITENOISE = 1
```

```
PRBS = 2
```

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000ASigGenTrigType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RISING** = 0

**FALLING** = 1

**GATE\_HIGH** = 2

**GATE\_LOW** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000ASigGenTrigSource(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**SCOPE\_TRIG** = 1

**AUX\_IN** = 2

**EXT\_IN** = 3

**SOFT\_TRIG** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AIndexMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000A_ThresholdMode(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**LEVEL** = 0

**WINDOW** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AThresholdDirection(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**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(value,
names=None,
*,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Bases: `IntEnum`

```
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(value,
names=None,
*,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Bases: `IntEnum`

`DONT_CARE = 0`

`TRUE = 1`

`FALSE = 2`

`MAX = 3`

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000ARatioMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`NONE = 0`

`AGGREGATE = 1`

`DECIMATE = 2`

`AVERAGE = 4`

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000APulseWidthType(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`NONE = 0`

`LESS_THAN = 1`

`GREATER_THAN = 2`

`IN_RANGE = 3`

`OUT_OF_RANGE = 4`

```
class msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldOffType(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**TIME = 0**

**MAX = 1**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000Channel(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`SQUARE = 0`

`TRIANGLE = 1`

`SINE = 2`

`MAX = 3`

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000TimeUnits(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000Info(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RISING** = 0

**FALLING** = 1

**MAX** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000OpenProgress(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**FAIL** = -1

**PENDING** = 0

**COMPLETE** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000EtsMode(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ThresholdDirection(value,
                                                                                   names=None,
                                                                                   *,
                                                                                   mod-
                                                                                   ule=None,
                                                                                   qual-
                                                                                   name=None,
                                                                                   type=None,
                                                                                   start=1,
                                                                                   bound-
                                                                                   ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**LEVEL** = 0

**WINDOW** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerState(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**DONT\_CARE** = 0

**TRUE** = 1

**FALSE** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000PulseWidthType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**LESS\_THAN** = 1

**GREATER\_THAN** = 2

**IN\_RANGE** = 3

**OUT\_OF\_RANGE** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ABandwidthLimiter(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`BW_FULL` = 0

`BW_20MHZ` = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex(value,
names=None,
*,
module=None,
qualified_name=None,
type=None,
start=1,
boundary=None)
```

Bases: `IntEnum`

`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(value,
names=None,
*, module=None,
qualified_name=None,
type=None,
start=1,
boundary=None)
```

Bases: `IntEnum`

`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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**PORT0** = 128

**PORT1** = 129

**PORT2** = 130

**PORT3** = 131

**MAX** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalChannel(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**AC** = 0

**DC** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelInfo(value,
                                                                           names=None,
                                                                           *,
                                                                           mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**RANGES = 0**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AEtsMode(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**OFF = 0**

**FAST = 1**

**SLOW = 2**

**MAX = 3**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ATimeUnits(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**FS = 0**

**PS = 1**

**NS = 2**



**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ASweepType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**WHITENOISE** = 1

**PRBS** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ASigGenTrigType(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RISING** = 0

**FALLING** = 1

**GATE\_HIGH** = 2

**GATE\_LOW** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ASigGenTrigSource(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              boundary=None)
```

Bases: `IntEnum`

**NONE** = 0

**SCOPE\_TRIG** = 1

**AUX\_IN** = 2

**EXT\_IN** = 3

**SOFT\_TRIG** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AIndexMode(value,
                                                                              names=None,
                                                                              *, module=
                                                                              None, qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              boundary=
                                                                              None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000A_ThresholdMode(value,
                                                                              names=None,
                                                                              *,
                                                                              module=
                                                                              None, qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              boundary=
                                                                              None)
```

Bases: `IntEnum`

`LEVEL = 0`

`WINDOW = 1`

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection(value,
names=None, *,
module=None,
qualified_name=None,
type=None,
start=1,
boundary=None)
```

Bases: `IntEnum`

`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(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`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(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`DONT_CARE = 0`

`TRUE = 1`

`FALSE = 2`

`MAX = 3`

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000ARatioMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**NONE = 0**

**AGGREGATE = 1**

**DECIMATE = 2**

**AVERAGE = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000APulseWidthType(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldOffType(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**TIME** = 0

**EVENT** = 1

**MAX** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex(value,
                                                                 names=None,
                                                                 *,
                                                                 mod-
                                                                 ule=None,
                                                                 qual-
                                                                 name=None,
                                                                 type=None,
                                                                 start=1,
                                                                 bound-
                                                                 ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                 names=None,
                                                                 *, mod-
                                                                 ule=None,
                                                                 qual-
                                                                 name=None,
                                                                 type=None,
                                                                 start=1,
                                                                 bound-
                                                                 ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

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(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RANGES = 0**

**RESISTANCES = 1**

**ACCELEROMETER = 2**

**PROBES = 3**

**TEMPERATURES = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000EtsMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000TimeUnits(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**FS** = 0

**PS** = 1

**NS** = 2

**US** = 3

**MS** = 4

**S** = 5

**MAX** = 6

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000SweepType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`UP = 0`

`DOWN = 1`

`UPDOWN = 2`

`DOWNUP = 3`

`MAX = 4`

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000OperationTypes(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`NONE = 0`

`WHITENOISE = 1`

`PRBS = 2`

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType(value,
                                                                           names=None,
                                                                           *, module=
                                                                           None, qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

`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(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RISING** = 0

**FALLING** = 1

**GATE\_HIGH** = 2

**GATE\_LOW** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000SigGenTrigSource(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**SCOPE\_TRIG** = 1

**AUX\_IN** = 2

**EXT\_IN** = 3

**SOFT\_TRIG** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000IndexMode(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdMode(value,
                                                                    names=None,
                                                                    *,
                                                                    module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**LEVEL** = 0

**WINDOW** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection(value,
                                                                    names=None,
                                                                    *,
                                                                    module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

```
DONT_CARE = 0
TRUE = 1
FALSE = 2
MAX = 3
```

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000RatioMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**AGGREGATE** = 1

**AVERAGE** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000PulseWidthType(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**LESS\_THAN** = 1

**GREATER\_THAN** = 2

**IN\_RANGE** = 3

**OUT\_OF\_RANGE** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000Ps4000HoldOffType(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`



**TIME** = 0

**MAX** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000FrequencyCounterRange(value,
names=None, *,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Bases: `IntEnum`

**FC\_2K** = 0

**FC\_20K** = 1

**FC\_20** = 2

**FC\_200** = 3

**FC\_MAX** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations(value,
names=None, *,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**WHITENOISE** = 1

**PRBS** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ABandwidthLimiter(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              quality=None,
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              boundary=None)
```

Bases: `IntEnum`

**BW\_FULL** = 0

**BW\_20KHZ** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ACoupling(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           quality=None,
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

**AC** = 0

**DC** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           quality=None,
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

**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(value,
names=None,
*,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**R\_315K** = 512

**R\_1100K** = 513

**R\_10M** = 514

**R\_MAX** = 3

**R\_ADCV** = 268435456

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AEtsMode(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ATimeUnits(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ASweepType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**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.PS4000AChannelLed(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**OFF** = 0

**RED** = 1

**GREEN** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AMetaType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**UNIT\_INFO** = 0

**DEVICE\_CAPABILITY** = 1

**DEVICE\_SETTINGS** = 2

**SIGNAL\_GENERATOR\_SETTINGS** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AMetaOperation(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**READ** = 0

**WRITE** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AMetaFormat(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**COMMA\_SEPERATED** = 0

**XML** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ASigGenTrigType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RISING** = 0

**FALLING** = 1

**GATE\_HIGH** = 2

**GATE\_LOW** = 3



```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ASigGenTrigSource(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**SCOPE\_TRIG** = 1

**AUX\_IN** = 2

**EXT\_IN** = 3

**SOFT\_TRIG** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AIndexMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdMode(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`LEVEL = 0`

`WINDOW = 1`

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection(value,
names=None,
*,
module=None,
qualified_name=None,
type=None,
start=1,
boundary=None)
```

Bases: `IntEnum`

`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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`DONT_CARE = 0`

`TRUE = 1`

`FALSE = 2`

`MAX = 3`

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ASensorState(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`CONNECT_STATE_FLOATING = 0`

`SENSOR_STATE_CONNECTED = 1`

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AFrequencyCounterRange(value,
                                                                                          names=None,
                                                                                          *,
                                                                                          mod-
                                                                                          ule=None,
                                                                                          qual-
                                                                                          name=None,
                                                                                          type=None,
                                                                                          start=1,
                                                                                          bound-
                                                                                          ary=None)
```

Bases: `IntEnum`

`FC_2K = 0`

**FC\_20K** = 1

**FC\_20** = 2

**FC\_200** = 3

**FC\_MAX** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AConditionsInfo(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**CLEAR** = 1

**ADD** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode(value,
                                                                              names=None,
                                                                              *, module=
                                                                              None, qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**AGGREGATE** = 1

**DECIMATE** = 2

**AVERAGE** = 4

**DISTRIBUTION** = 8

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000APulseWidthType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**LESS\_THAN** = 1

**GREATER\_THAN** = 2

**IN\_RANGE** = 3

**OUT\_OF\_RANGE** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelInfo(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RANGES** = 0

**RESISTANCES** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                                   names=None,
                                                                                   *,
                                                                                   mod-
                                                                                   ule=None,
                                                                                   qual-
                                                                                   name=None,
                                                                                   type=None,
                                                                                   start=1,
                                                                                   bound-
                                                                                   ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: [IntEnum](#)

**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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000TimeUnits(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**FS** = 0

**PS** = 1

**NS** = 2

**US** = 3

**MS** = 4

**S** = 5

**MAX** = 6

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000SweepType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**UP** = 0

**DOWN** = 1

**UPDOWN** = 2

**DOWNUP = 3**

**MAX = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH** = 2

**GATE\_LOW** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000SigGenTrigSource(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**SCOPE\_TRIG** = 1

**AUX\_IN** = 2

**EXT\_IN** = 3

**SOFT\_TRIG** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000IndexMode(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdMode(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**LEVEL = 0**

**WINDOW = 1**

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ThresholdDirection(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**DONT\_CARE** = 0

**TRUE** = 1

**FALSE** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000RatioMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**AGGREGATE** = 1

**DECIMATE** = 2

**AVERAGE** = 4

**DISTRIBUTION** = 8

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000PulseWidthType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`NONE = 0`

`LESS_THAN = 1`

`GREATER_THAN = 2`

`IN_RANGE = 3`

`OUT_OF_RANGE = 4`

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelInfo(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`RANGES = 0`

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ADeviceResolution(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`RES_8BIT = 0`

`RES_12BIT = 1`

`RES_14BIT = 2`

`RES_15BIT = 3`

`RES_16BIT = 4`



```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AExtraOperations(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**WHITENOISE** = 1

**PRBS** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ABandwidthLimiter(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**BW\_FULL** = 0

**BW\_20MHZ** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ACoupling(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**AC** = 0

**DC** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ATimeUnits(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**FS** = 0

**PS** = 1

**NS** = 2

**US** = 3

**MS** = 4

**S** = 5

**MAX** = 6

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ASweepType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**UP** = 0

**DOWN** = 1

**UPDOWN** = 2

**DOWNUP** = 3

**MAX** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**RISING = 0**

**FALLING = 1**

**GATE\_HIGH = 2**

**GATE\_LOW = 3**

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ASigGenTrigSource(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**SCOPE\_TRIG** = 1

**AUX\_IN** = 2

**EXT\_IN** = 3

**SOFT\_TRIG** = 4

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AIndexMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdMode(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**LEVEL** = 0

**WINDOW** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection(value,
names=None, *,
module=None,
qualified_name=None,
type=None,
start=1,
boundary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**DONT\_CARE** = 0

**TRUE** = 1

**FALSE** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ATriggerWithinPreTrigger(value,
                                                                                          names=
                                                                                          *,
                                                                                          mod-
                                                                                          ule=None,
                                                                                          qual-
                                                                                          name=
                                                                                          type=None,
                                                                                          start=1,
                                                                                          bound-
                                                                                          ary=None)
```

Bases: `IntEnum`

**DISABLE** = 0

**ARM** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE** = 0

**AGGREGATE** = 1



**DECIMATE = 2**

**AVERAGE = 4**

**DISTRIBUTION = 8**

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000APulseWidthType(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelInfo(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**RANGES = 0**

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000ExternalFrequency(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**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(value,
                                         names=None,
                                         *,
                                         module=None,
                                         qual-
                                         name=None,
                                         type=None,
                                         start=1,
                                         bound-
                                         ary=None)
```

Bases: `IntEnum`

**BW\_FULL** = 0

**BW\_20MHZ** = 1

**BW\_25MHZ** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000Channel(value,
                               names=None,
                               *, module=
                               None, qual-
                               name=None,
                               type=None,
                               start=1,
                               bound-
                               ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                                   names=None,
                                                                                   *,
                                                                                   mod-
                                                                                   ule=None,
                                                                                   qual-
                                                                                   name=None,
                                                                                   type=None,
                                                                                   start=1,
                                                                                   bound-
                                                                                   ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                       names=None,
                                                                       *,
                                                                       module=None,
                                                                       qual-
                                                                       name=None,
                                                                       type=None,
                                                                       start=1,
                                                                       bound-
                                                                       ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**AC** = 0

**DC\_1M** = 1

**DC\_50R** = 2

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000EtsMode(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**OFF** = 0

**FAST** = 1

**SLOW** = 2

**MAX = 3**

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000TimeUnits(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**FS = 0**

**PS = 1**

**NS = 2**

**US = 3**

**MS = 4**

**S = 5**

**MAX = 6**

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000SweepType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**UP = 0**

**DOWN = 1**

**UPDOWN = 2**

**DOWNUP = 3**

**MAX = 4**

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000WaveType(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)

Bases: IntEnum

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(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)

Bases: IntEnum

OFF = 0

WHITENOISE = 1

PRBS = 2
```

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000SigGenTrigType(value,
                                     names=None,
                                     *,
                                     module=None,
                                     qual-
                                     name=None,
                                     type=None,
                                     start=1,
                                     boundary=None)

Bases: IntEnum

RISING = 0
FALLING = 1
GATE_HIGH = 2
GATE_LOW = 3

class msl.equipment.resources.picotech.picoscope.enums.PS6000SigGenTrigSource(value,
                                         names=None,
                                         *,
                                         module=None,
                                         qual-
                                         name=None,
                                         type=None,
                                         start=1,
                                         boundary=None)

Bases: IntEnum

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(value,
                                   names=None,
                                   *, module=None,
                                   qual-
                                   name=None,
                                   type=None,
                                   start=1,
                                   boundary=None)
```

Bases: `IntEnum`

**SINGLE** = 0

**DUAL** = 1

**QUAD** = 2

**MAX** = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdMode(value,
                                     names=None,
                                     *,
                                     module=None,
                                     qual-
                                     name=None,
                                     type=None,
                                     start=1,
                                     bound-
                                     ary=None)
```

Bases: `IntEnum`

**LEVEL** = 0

**WINDOW** = 1

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000ThresholdDirection(value,
                                           names=None,
                                           *,
                                           module=None,
                                           qual-
                                           name=None,
                                           type=None,
                                           start=1,
                                           bound-
                                           ary=None)
```

Bases: `IntEnum`

**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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: [IntEnum](#)

DONT\_CARE = 0

TRUE = 1

FALSE = 2

MAX = 3

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000RatioMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: [IntEnum](#)

NONE = 0

**AGGREGATE = 1**

**AVERAGE = 2**

**DECIMATE = 4**

**DISTRIBUTION = 8**

```
class msl.equipment.resources.picotech.picoscope.enums.PS6000PulseWidthType(value,  
                                                                           names=None,  
                                                                           *,  
                                                                           mod-  
                                                                           ule=None,  
                                                                           qual-  
                                                                           name=None,  
                                                                           type=None,  
                                                                           start=1,  
                                                                           bound-  
                                                                           ary=None)
```

Bases: `IntEnum`

**NONE = 0**

**LESS\_THAN = 1**

**GREATER\_THAN = 2**

**IN\_RANGE = 3**

**OUT\_OF\_RANGE = 4**

## **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(path)`

Parse a PicoScope header file.

**Parameters**

**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_dict)`

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(header_dict, pi-costatus_h_path)`

Creates the `_picoscope_enums.py` file

`msl.equipment.resources.picotech.picoscope.helper.create_picoscope_structs_file(header_dict)`

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_file(header_dict)`

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(header_dict)`

`msl.equipment.resources.picotech.picoscope.helper.print_common_functions(header_dict, 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:** 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: *ConnectionSDK*

Use the PicoScope SDK to communicate with the oscilloscope.

The *properties* for a PicoScope connection supports the following key-value pairs in the *Connections Database*:

```
'open': bool, Whether to open the connection synchronously [default: True]
'open_async': bool, Whether to open the connection asynchronously
↳ [default: False]
'auto_select_power': bool, For devices that can be powered by an AC
↳ adaptor or a USB cable [default: True]
'resolution': str, Only valid for ps5000a [default: '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*

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

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

**property handle**

Returns the handle to the SDK library.

**Type**

`int`

**property channel**

The information about each channel.

**Type**

`dict of PicoScopeChannel`

**property dt**

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

**Type**

`float`

**property pre\_trigger**

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

**Type**

`float`

**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** (*PicoScopeInfoApi*, *PS2000Info* or *PS3000Info*, optional) – An enum value, or if `None` then request all information from the PicoScope. The enum depends on the model number of the PicoScope that you are connected to.
- **include\_name** (*bool*, optional) – If `True` then includes the enum member name as a prefix. For example, returns 'CAL\_DATE: 09Aug16' if *include\_name* is `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*, optional) – The number of seconds before the trigger event to start acquiring data.
- **segment\_index** (*int*, optional) – Specifies which memory segment to save the data to (see manual).
- **callback** (*BlockReady*, optional) – 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*, optional) – The number of seconds before the trigger event to start acquiring data.
- **auto\_stop** (*bool*, optional) – A flag that specifies if the streaming should stop when all of samples have been captured.
- **factor** (*int*, optional) – The down-sampling factor that will be applied to the raw data.
- **ratio\_mode** (*enum.IntEnum*, optional) – 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`, optional) – The impedance and coupling type.
- **scale** (`enum.IntEnum`, optional) – The input voltage range.
- **offset** (`float`, optional) – 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`, optional) – The bandwidth limiter to use.
- **enabled** (`bool`, optional) – 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`, optional) – Which memory segment to save the data to.
- **oversample** (`int`, optional) – The amount of over-sample required.

#### Returns

- `float` – The sampling interval, i.e. *dt*.
- `int` – The number of samples that will be acquired.

#### Raises

**PicoTechError** – 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`, optional) – The time, in seconds, between the trigger occurring and the first sample.
- **direction** (`enum.IntEnum`, optional) – The direction in which the signal must move to cause a trigger.
- **timeout** (`float`, optional) – 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`, optional) – 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, which are found in the `structs` module.

**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\_ptrs*)

Bases: `PicoScope`

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

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

**Parameters**

- **record** (*EquipmentRecord*) – A 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\(\)](#).

**run\_streaming\_ns**(*sample\_interval*, *time\_units*, *max\_samples*, *auto\_stop*,  
*no\_of\_samples\_per\_aggregate*, *overview\_buffer\_size*)

This function tells the scope unit to start collecting data in fast streaming mode. The function returns immediately without waiting for data to be captured. After calling this function, you should next call [get\\_streaming\\_last\\_values\(\)](#) to copy the data to your application's buffer.

**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 `TriggerConditions` structures, which are found in the `structs` module. 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: `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 `connect()` method to connect to the equipment.

### Parameters

- **record** (*EquipmentRecord*) – A 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](#), optional) – 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 [StreamingReady 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`, optional) – The number of samples required. If `None` then automatically determine the number of samples to retrieve.
- **start\_index** (`int`, optional) – 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`, optional) – The down-sampling factor that will be applied to the raw data.
- **ratio\_mode** (`enum.IntEnum`, optional) – Which down-sampling mode to use. A `RatioMode` enum.
- **segment\_index** (`int`, optional) – 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** (*callback*) – A *DataReady callback* function.
- **num\_samples** (`int`, optional) – The number of samples required. If `None` then automatically determine the number of samples to retrieve.
- **start\_index** (`int`, optional) – 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`, optional) – The downsampling factor that will be applied to the raw data.
- **ratio\_mode** (`enum.IntEnum`, optional) – Which down-sampling mode to use. A `RatioMode` enum.
- **segment\_index** (`int`, optional) – 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`, optional) – A int16, numpy array. If `None` then use a pre-allocated array.
- **mode** (`enum.IntEnum`, optional) – An enum value or member name from `RatioMode`.
- **segment\_index** (`int`, optional) – 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.c_longlong`) – An array of 64-bit words (`ctypes.c_int64`), 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, repetition\_rate=None, offset\_voltage=0.0, pk\_to\_pk=None, start\_delta\_phase=None, stop\_delta\_phase=None, delta\_phase\_increment=0, dwell\_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.

#### Parameters

- **waveform** (`numpy.ndarray`) – The arbitrary waveform, in volts.
- **repetition\_rate** (`float`, optional) – The requested repetition rate (frequency) of the entire arbitrary waveform. The actual repetition rate that is used may be different based on the specifications of the AWG. If specified then the `sig_gen_frequency_to_phase()` method is called to determine the value of `start_delta_phase`.
- **offset\_voltage** (`float`, optional) – The voltage offset, in volts, to be applied to the waveform.
- **pk\_to\_pk** (`float`, optional) – The peak-to-peak voltage, in volts, of the waveform signal. If `None` then uses the maximum value of the waveform to determine the peak-to-peak voltage.
- **start\_delta\_phase** (`int`, optional) – The initial value added to the phase accumulator as the generator begins to step through the waveform buffer.
- **stop\_delta\_phase** (`int`, optional) – The final value added to the phase accumulator before the generator restarts or reverses the sweep. When frequency sweeping is not required, set equal to `start_delta_phase`.
- **delta\_phase\_increment** (`int`, optional) – The amount added to the delta phase value every time the `dwell_count` period expires. This determines the amount by which the generator sweeps the output frequency in each dwell period. When frequency sweeping is not required, set to zero.
- **dwell\_count** (`int`, optional) – The time, in units of `dacPeriod`, between successive additions of `delta_phase_increment` to the delta phase accumulator. This determines the rate at which the generator sweeps the output frequency.
- **sweep\_type** (`enum.IntEnum`, optional) – 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`, optional) – The type of waveform to be produced, specified by one of the following enumerated types (B models only): OFF, WHITENOISE, PRBS
- **index\_mode** (`enum.IntEnum`, optional) – Specifies how the signal will be formed from the arbitrary waveform data. Possible values are SINGLE or DUAL.
- **shots** (`int`, optional) – If `None` then start and run continuously after trigger occurs.
- **sweeps** (`int`, optional) – If `None` then start a sweep and continue after trigger occurs.
- **trigger\_type** (`enum.IntEnum`, optional) – The type of trigger that will be applied to the signal generator. One of: RISING, FALLING, GATE\_HIGH, GATE\_LOW.
- **trigger\_source** (`enum.IntEnum`, optional) – The source that will trigger the signal generator. If `None` then run without waiting for trigger.
- **ext\_in\_threshold** (`int`, optional) – Used to set trigger level for external trigger.

#### Returns

`numpy.ndarray` – The arbitrary waveform, in ADU.

#### Raises

`PicoTechError` – If the value of an input parameter is invalid.

**set\_sig\_gen\_builtin**(*offset\_voltage, pk\_to\_pk, wave\_type, start\_frequency, stop\_frequency, increment, dwell\_time, sweep\_type, operation, 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. Call `set_sig_gen_builtin_v2()` instead, which uses double-precision arguments.

**set\_sig\_gen\_builtin\_v2**(*offset\_voltage=0.0, pk\_to\_pk=1.0, wave\_type='sine', start\_frequency=1.0, stop\_frequency=None, increment=0.1, dwell\_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_builtin()` with double-precision frequency arguments for more precise control at low frequencies.

This function is invalid for ps4000 and ps4000a.

#### Parameters

- **offset\_voltage** (`float`, optional) – The voltage offset, in volts, to be applied to the waveform.
- **pk\_to\_pk** (`float`, optional) – The peak-to-peak voltage, in volts, of the waveform signal.
- **wave\_type** (`enum.IntEnum`, optional) – The type of waveform to be generated. A WaveType enum.

- **start\_frequency** (*float*, optional) – The frequency that the signal generator will initially produce.
- **stop\_frequency** (*float*, optional) – The frequency at which the sweep reverses direction or returns to the initial frequency.
- **increment** (*float*, optional) – The amount of frequency increase or decrease in sweep mode.
- **dwelt\_time** (*float*, optional) – The time, in seconds, for which the sweep stays at each frequency.
- **sweep\_type** (*enum.IntEnum*, optional) – 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*, optional) – The type of waveform to be produced, specified by one of the following enumerated types (B models only): OFF, WHITENOISE, PRBS
- **shots** (*int*, optional) – If *None* then start and run continuously after trigger occurs.
- **sweeps** (*int*, optional) – If *None* then start a sweep and continue after trigger occurs.
- **trigger\_type** (*enum.IntEnum*, optional) – The type of trigger that will be applied to the signal generator. One of: RISING, FALLING, GATE\_HIGH, GATE\_LOW.
- **trigger\_source** (*enum.IntEnum*, optional) – The source that will trigger the signal generator. If *None* then run without waiting for trigger.
- **ext\_in\_threshold** (*int*, optional) – 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\_builtin**(*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**(*repetition\_rate, 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

- **repetition\_rate** ([float](#)) – The requested repetition rate (frequency) of the entire arbitrary waveform.
- **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.

#### Raises

[PicoTechError](#) – If the value of an input parameter is invalid.

**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 TriggerConditions structures, which are found in the [structs](#) module, 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 [TriggerChannelProperties structs](#)) – A list of [TriggerChannelProperties](#) structures describing the requested properties.
- **timeout** ([float](#), optional) – 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](#), optional) – 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: [PicoScope2k3k](#)

A wrapper around the PicoScope ps2000 SDK.

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

#### Parameters

**record** ([EquipmentRecord](#)) – A 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*, *dwel\_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*, *dwel\_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: *PicoScopeApi*

A wrapper around the PicoScope ps2000a SDK.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A 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: *PicoScope2k3k*

A wrapper around the PicoScope ps3000 SDK.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

### **Parameters**

**record** (*EquipmentRecord*) – A 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: *PicoScopeApi*

A wrapper around the PicoScope ps3000a SDK.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

### **Parameters**

**record** (*EquipmentRecord*) – A 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.

#### Returns

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: [PicoScopeApi](#)

A wrapper around the PicoScope ps4000 SDK.

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

**Parameters**

**record** ([EquipmentRecord](#)) – A 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: [PicoScopeApi](#)

A wrapper around the PicoScope ps4000a SDK.

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A 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 = 80000000.0
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: [`PicoScopeApi`](#)

A wrapper around the PicoScope ps5000 SDK.

Do not instantiate this class directly. Use the [`connect\(\)`](#) method to connect to the equipment.

### **Parameters**

**record** ([`EquipmentRecord`](#)) – A 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 = 200000000.0
SQUARE_MAX_FREQUENCY = 200000000.0
TRIANGLE_MAX_FREQUENCY = 200000000.0
SINC_MAX_FREQUENCY = 200000000.0
RAMP_MAX_FREQUENCY = 200000000.0
HALF_SINE_MAX_FREQUENCY = 200000000.0
GAUSSIAN_MAX_FREQUENCY = 200000000.0
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: *PicoScopeApi*

A wrapper around the PicoScope ps5000a SDK.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

#### **Parameters**

**record** (*EquipmentRecord*) – A 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**

*PS5000ADeviceResolution* – 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: *PicoScopeApi*

A wrapper around the PicoScope ps6000 SDK.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A 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.  
**PS2000TriggerChannelProperties**

Bases: 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: 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: 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: 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: 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.  
**PS2000ADigitalChannelDirections**

Bases: Structure

**channel**

Structure/Union member

**direction**

Structure/Union member

**class** msl.equipment.resources.picotech.picoscope.structs.  
**PS2000ATriggerChannelProperties**

Bases: 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: 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: 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: 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: 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.PS3000ATriggerConditionsV2`

Bases: 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: 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: 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.PS3000ADigitalChannelDirections`

Bases: Structure

**channel**

Structure/Union member

**direction**

Structure/Union member

**class** `msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerChannelProperties`

Bases: 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: 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: 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: 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: 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: Structure

**channel**  
Structure/Union member

**state**  
Structure/Union member

**class** msl.equipment.resources.picotech.picoscope.structs.**PS4000ADirection**

Bases: Structure

**channel**  
Structure/Union member

**direction**

Structure/Union member

**class** msl.equipment.resources.picotech.picoscope.structs.PS4000ACondition

Bases: Structure

**condition**

Structure/Union member

**source**

Structure/Union member

**class** msl.equipment.resources.picotech.picoscope.structs.  
**PS4000ATriggerChannelProperties**

Bases: 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: Structure

**channel**

Structure/Union member

**state**

Structure/Union member

**class**  
msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerConditions

Bases: 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: 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: 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: 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: 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: 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.  
**PS5000ATriggerChannelProperties**

Bases: 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: 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: 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.  
**PS6000TriggerChannelProperties**

Bases: 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.picotech.pt104 module**

Pico Technology PT-104 Platinum Resistance Data Logger.

**class** msl.equipment.resources.picotech.pt104.Pt104DataType(*value, names=None, \*,  
module=None,  
qualname=None,  
type=None, start=1,  
boundary=None*)

Bases: `IntEnum`

The allowed data types for a PT-104 Data Logger.

**OFF** = 0

**PT100** = 1

**PT1000** = 2

**RESISTANCE\_TO\_375R** = 3

**RESISTANCE\_TO\_10K** = 4

**DIFFERENTIAL\_TO\_115MV** = 5

**DIFFERENTIAL\_TO\_2500MV** = 6

**SINGLE\_ENDED\_TO\_115MV** = 7

**SINGLE\_ENDED\_TO\_2500MV** = 8

**MAX\_DATA\_TYPES** = 9

`msl.equipment.resources.picotech.pt104.enumerate_units(comm_type='all')`

Find PT-104 Platinum Resistance Data Logger's.

This routine returns a list of all the attached PT-104 devices of the specified communication type.

---

**Note:** You cannot call this function after you have opened a connection to a Data Logger.

---

#### Parameters

**comm\_type** (`str`, optional) – The communication type used by the PT-104. Can be any of the following values: 'usb', 'ethernet', 'enet', 'all'

#### Returns

`list` of `str` – A list of serial numbers of the PT-104 Data Loggers that were found.

**class** `msl.equipment.resources.picotech.pt104.PT104(record)`

Bases: `ConnectionSDK`

Uses the PicoTech SDK to communicate with a PT-104 Platinum Resistance Data Logger.

The *properties* for a PT-104 connection supports the following key-value pairs in the *Connections Database*:

`'ip_address': str`, The IP address **and** port number of the PT-104 (e.g.,  
↪ `'192.168.1.201:1234'`)  
`'open_via_ip': bool`, Whether to connect to the PT-104 by Ethernet. ↪  
↪ Default **is** to connect by USB.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**MIN\_WIRES** = 2

**MAX\_WIRES** = 4

**DataType**

alias of *Pt104DataType*

**disconnect()**

Disconnect from the PT-104 Data Logger.

**get\_ip\_details()**

Reads the IP details of the PT-104 Data Logger.

**Returns**

*dict* – The IP details.

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

Retrieves information about the PT-104 Data Logger.

If the device fails to open, or no device is opened only the driver version is available.

**Parameters**

- **info** (*PicoScopeInfoApi*, optional) – An enum value or member name. If *None* then request all information from the PT-104.
- **include\_name** (*bool*, optional) – If *True* then includes the enum member name as a prefix. For example, returns 'CAL\_DATE: 09Aug16' if *include\_name* is *True* else '09Aug16'.

**Returns**

*str* – The requested information from the PT-104 Data Logger.

**get\_value**(*channel, filtered=False*)

Get the most recent reading for the specified channel.

Once you open the driver and define some channels, the driver begins to take continuous readings from the PT-104 Data Logger.

The scaling of measurements is as follows:

| Range                 | Scaling              |
|-----------------------|----------------------|
| Temperature           | value * 1/1000 deg C |
| Voltage (0 to 2.5 V)  | value * 10 nV        |
| Voltage (0 to 115 mV) | value * 1 nV         |
| Resistance            | value * 1 mOhm       |

**Parameters**

- **channel** (*int*) – The number of the channel to read, from 1 to 4 in differential mode or 1 to 8 in single-ended mode.

- **filtered** (*bool*, optional) – If set to *True*, the driver returns a low-pass filtered value of the temperature. The time constant of the filter depends on the channel parameters as set by *set\_channel()*, and on how many channels are active.

**Returns**

*float* – The latest reading for the specified channel.

**open()**

Open the connection to the PT-104 via USB.

**open\_via\_ip(address=None)**

Open the connection to the PT-104 via ETHERNET.

**Parameters**

**address** (*str*, optional) – The IP address and port number to use to connect to the PT-104. For example, '192.168.1.201:1234'. If *None* then uses the 'ip\_address' value of the *properties*

**set\_channel(channel, data\_type, num\_wires)**

Configure a single channel of the PT-104 Data Logger.

The fewer channels selected, the more frequently they will be updated. Measurement takes about 1 second per active channel.

If a call to the *set\_channel()* method has a data type of single-ended, then the specified channel's 'sister' channel is also enabled. For example, enabling 3 also enables 7.

**Parameters**

- **channel** (*int*) – The channel you want to set the details for. It should be between 1 and 4 if using single-ended inputs in voltage mode.
- **data\_type** (*DataType*) – The type of reading you require. Can be an enum value or member name.
- **num\_wires** (*int*) – The number of wires the PT100 or PT1000 sensor has (2, 3 or 4)

**set\_ip\_details(enabled, ip\_address=None, port=None)**

Writes the IP details to the device.

**Parameters**

- **enabled** (*bool*) – Whether to enable or disable Ethernet communication for this device.
- **ip\_address** (*str*, optional) – The new IP address. If *None* then do not change the IP address.
- **port** (*int*, optional) – The new port number. If *None* then do not change the port number.

**set\_mains(hertz)**

Inform the driver of the local mains (line) frequency.

This helps the driver to filter out electrical noise.

**Parameters**

**hertz** (*int*) – Either 50 or 60.

**msl.equipment.resources.princeton\_instruments package**

Wrapper around the `ARC_Instrument.dll` SDK from Princeton Instruments.

**Submodules****msl.equipment.resources.princeton\_instruments.arc\_instrument module**

Wrapper around the `ARC_Instrument.dll` SDK from Princeton Instruments.

The wrapper was written using v2.0.3 of the SDK.

**class** `msl.equipment.resources.princeton_instruments.arc_instrument.PrincetonInstruments`(*record*)

Bases: *Connection*

Wrapper around the `ARC_Instrument.dll` SDK from Princeton Instruments.

The *properties* for a `ARCInstrument` connection supports the following key-value pairs in the *Connections Database*:

`'sdk_path': str`, The path to the SDK [default: `'ARC_Instrument_x64.dll'`]  
`'open': bool`, Whether to automatically *open* the connection [default: *True*]

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**disconnect()**

Close all open connections.

**static close\_enum(enum)**

Function to close an open enumeration.

**Parameters**

**enum** (*int*) – A handle defined in `ARC_Open_xxxx` by which the instrument is to be addressed.

**det\_read(det\_num)**

Readout a single detector.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*float* – Reading value for the selected detector.

**det\_readall()**

Readout all detectors.

**Returns**

**tuple** of **float** – The reading of each detector.

**det\_start\_nonblock\_read**(*det\_num*)

Start to readout a single detector, but return immediately (non-blocking read).

**Parameters**

**det\_num** (**int**) – The detector to be addressed.

**det\_nonblock\_read\_done**(*det\_num*)

Returns the detector value.

**Parameters**

**det\_num** (**int**) – The detector to be addressed.

**Returns**

**float** – The value of the detector.

**filter\_home**()

Homes the filter wheel.

**mono\_filter\_home**()

Homes the filter wheel.

**mono\_grating\_calc\_gadjust**(*grating*, *wave*, *ref\_wave*)

Calculate a new grating GAdjust.

**Parameters**

- **grating** (**int**) – The number of the grating that is to be addressed.
- **wave** (**float**) – The wavelength, in nm, on which a peak is currently falling.
- **ref\_wave** (**float**) – The wavelength, in nm, on which the peak should fall.

**Returns**

**int** – The calculated grating GAdjust. Note - this value is specific to a specific instrument-grating combination and not transferable between instruments.

**mono\_grating\_calc\_offset**(*grating*, *wave*, *ref\_wave*)

Calculate a new grating offset.

**Parameters**

- **grating** (**int**) – The number of the grating that is to be addressed.
- **wave** (**float**) – The wavelength, in nm, on which a peak is currently falling.
- **ref\_wave** (**float**) – The wavelength, in nm, on which the peak should fall.

**Returns**

**int** – The calculated grating offset. Note - this value is specific to a specific instrument-grating combination and not transferable between instruments.

**mono\_grating\_install**(*grating, density, blaze, nm\_blaze, um\_blaze, hol\_blaze, mirror, reboot*)

Install a new grating.

**Parameters**

- **grating** (*int*) – The number of the grating that is to be addressed.
- **density** (*int*) – The groove density in grooves per mm of the grating.
- **blaze** (*str*) – The Blaze string.
- **nm\_blaze** (*bool*) – Whether the grating has a nm blaze.
- **um\_blaze** (*bool*) – Whether the grating has a um blaze.
- **hol\_blaze** (*bool*) – Whether the grating has a holographic blaze.
- **mirror** (*bool*) – Whether the grating position is a mirror.
- **reboot** (*bool*) – Whether to reboot the monochromator after installing.

**mono\_grating\_uninstall**(*grating, reboot*)

Uninstall a grating.

**Parameters**

- **grating** (*int*) – The number of the grating that is to be uninstalled.
- **reboot** (*bool*) – Whether to reboot the monochromator after uninstalling.

**mono\_move\_steps**(*num\_steps*)

Move the grating a set number of steps.

**Parameters**

- **num\_steps** (*int*) – Number of steps to move the wavelength drive.

**mono\_reset**()

Reset the monochromator.

**mono\_restore\_factory\_settings**()

Restore the instrument factory settings.

**mono\_scan\_done**()

Check if a scan has completed.

**Returns**

- *bool* – Whether the scan finished (the scan ended or the grating wavelength limit was reached).
- *float* – The current wavelength, in nm, of the grating.

**mono\_slit\_home**(*slit\_num*)

Homes a motorized slit.

**Parameters**

- **slit\_num** (*int*) – The slit to be homed.
- 1 - Side entrance slit.

- 2 - Front entrance slit.
- 3 - Front exit slit.
- 4 - Side exit slit.
- 5 - Side entrance slit on a double slave unit.
- 6 - Front entrance slit on a double slave unit.
- 7 - Front exit slit on a double slave unit.
- 8 - Side exit slit on a double slave unit.

**static mono\_slit\_name**(*slit\_num*)

Returns a text description of a slit position.

**Parameters**

**slit\_num** (*int*) – The slit to be addressed.

**Returns**

*str* – The description of the slit port location.

**mono\_start\_jog**(*jog\_max\_rate*, *jog\_forwards*)

Start jogging the wavelength of the monochromator.

**Parameters**

- **jog\_max\_rate** (*bool*) – Whether to jog at the max scan rate or at the current scan rate.
  - *False* - Current Scan Rate.
  - *True* - Max Scan Rate.
- **jog\_forwards** (*bool*) – Whether to jog forward (increasing nm) or backwards (decreasing nm).
  - *False* - Jog Backwards (decreasing nm).
  - *True* - Jog Forwards (increasing nm).

**mono\_start\_scan\_to\_nm**(*wavelength\_nm*)

Start a wavelength scan.

**Parameters**

**wavelength\_nm** (*float*) – Wavelength in nm we are to scan to. Note - the value should not be lower than the current wavelength and should not exceed the current grating wavelength limit.

**mono\_stop\_jog**()

End a wavelength jog.

**calc\_mono\_slit\_bandpass**(*slit\_num*, *width*)

Calculates the band pass for the provided slit width.

**Parameters**

- **slit\_num** (*int*) – The slit number.
  - 1 - Side entrance slit.
  - 2 - Front entrance slit.



- 3 - Front exit slit.
- 4 - Side exit slit.
- 5 - Side entrance slit on a double slave unit.
- 6 - Front entrance slit on a double slave unit.
- 7 - Front exit slit on a double slave unit.
- 8 - Side exit slit on a double slave unit.
- **width** (*float*) – The slit width that the band pass is being calculated for.

**Returns**

*float* – The calculated band pass for the slit, in nm.

**ncl\_filter\_home()**

Home the filter wheel.

**open\_filter**(*enum\_num*)

Function to open a Filter Wheel.

**Parameters**

**enum\_num** (*int*) – Number in the enumeration list to open.

**open\_mono**(*enum\_num*)

Function to open a Monochromator.

**Parameters**

**enum\_num** (*int*) – Number in the enumeration list to open.

**open\_mono\_port**(*port*)

Function to open a Monochromator on a specific COM Port.

**Parameters**

**port** (*int* or *str*) – The COM port (e.g., 5 or 'COM5').

**open\_mono\_serial**(*serial*)

Function to open a Monochromator with a specific serial number.

**Parameters**

**serial** (class:*int* or *str*) – The serial number of the Monochromator.

**open\_filter\_port**(*port*)

Function to open a Filter Wheel on a specific COM Port.

**Parameters**

**port** (*int* or *str*) – The COM port (e.g., 5 or 'COM5').

**open\_filter\_serial**(*serial*)

Function to open a Filter Wheel with a specific serial number.

**Parameters**

**serial** (class:*int* or *str*) – The serial number of the Filter Wheel.

**open\_readout**(*port\_num*)

Function to open a Readout System (NCL/NCL-Lite).

**Parameters**

**port\_num** (*int*) – Number in the enumeration list to open.

**Returns**

- *int* – If a mono is attached to an NCL on port 1, then this is the enum by which to address the first mono.
- *int* – If a mono is attached to an NCL on port 2, then this is the enum by which to address the second mono.

**open\_readout\_port**(*port*)

Function to open a Readout System (NCL/NCL-Lite) on a specific COM Port.

**Parameters**

**port** (class:*int* or *str*) – The COM port (e.g., 5 or 'COM5').

**Returns**

- *int* – If a mono is attached to an NCL on port 1, then this is the enum by which to address the first mono.
- *int* – If a mono is attached to an NCL on port 2, then this is the enum by which to address the second mono.

**static search\_for\_inst**()

Function used to search for all attached Acton Research Instruments.

**Returns**

*int* – The number of Acton Research Instruments found and enumerated. Enumeration list starts with zero and ends with the “number found” minus one.

**valid\_det\_num**(*det\_num*)

Check if the detector number is valid on the Readout System.

**Parameters**

**det\_num** (*int*) – The detector number to check.

**Returns**

*bool* – Whether the detector number is valid on the Readout System.

**static valid\_enum**(*enum*)

Check if an enumeration is valid and the instrument is open.

**Parameters**

**enum** (*int*) – The enumeration to check.

**Returns**

*bool* – Whether the enumeration is valid and the instrument is open.

**static valid\_mono\_enum**(*mono\_enum*)

Check if a monochromator enumeration is valid and the instrument is open.

All functions call this function to verify that the requested action is valid.

**Parameters**

**mono\_enum** (*int*) – The enumeration to check.

**Returns**

`bool` – Whether the enumeration is valid and the instrument is open.

**static valid\_readout\_enum**(*readout\_enum*)

Check if a Readout System (NCL/NCL-Lite) enumeration is valid and the instrument is open.

**Parameters**

**readout\_enum** (`int`) – The enumeration to check.

**Returns**

`bool` – Whether the enumeration is valid and the instrument is open.

**static valid\_filter\_enum**(*filter\_enum*)

Check if a Filter Wheel enumeration is valid and the instrument is open.

**Parameters**

**filter\_enum** (`int`) – The enumeration to check.

**Returns**

`bool` – Whether the enumeration is valid and the instrument is open.

**static ver**()

Get the DLL version number.

It is the only function that can be called at any time.

**Returns**

- `int` – The major version number.
- `int` – The minor version number.
- `int` – The build version number.

**get\_det\_bipolar**(*det\_num*)

Return if a detector takes bipolar (+/-) readings.

**Parameters**

**det\_num** (`int`) – The detector to be addressed.

**Returns**

`bool` – `True` if the detector is bipolar (+/-).

**get\_det\_bipolar\_str**(*det\_num*)

Return the description of the detector polarity.

**Parameters**

**det\_num** (`int`) – The detector to be addressed.

**Returns**

`str` – A description of whether the detector is uni-polar or bipolar.

**get\_det\_hv\_volts**(*det\_num*)

Return the high voltage volts setting.

**Parameters**

**det\_num** (`int`) – The detector to be addressed.

**Returns**

`int` – High voltage volts that the detector is set to.

**get\_det\_hv\_on**(*det\_num*)

Return if the high voltage for a detector is turned on.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*bool* – Whether the high voltage is turned on.

**get\_det\_num\_avg\_read**()

Return the number of readings that are averaged.

**Returns**

*int* – Number of readings averaged into a single reading.

**get\_det\_range**(*det\_num*)

Return the detector range factor.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*int* – The detector gain range.

- Detector Range 0 - 1x
- Detector Range 1 - 2x
- Detector Range 2 - 4x
- Detector Range 3 - 50x
- Detector Range 4 - 200x

**get\_det\_range\_factor**(*det\_num*)

Return the detector range multiplier.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*int* – The detector range multiplier.

**get\_det\_type**(*det\_num*)

Return the detector readout type (Current, Voltage, Photon Counting).

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*int* –

The detector readout type.

- Detector Type 1 - Current
- Detector Type 2 - Voltage
- Detector Type 3 - Photon Counting

**get\_det\_type\_str**(*det\_num*)

Return a description of the detector readout type.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*str* – The detector readout type (Current, Voltage, Photon Counting).

**get\_filter\_max\_pos**()

Returns the maximum filter position.

**Returns**

*int* – Returns the maximum position possible with the filter wheel.

**get\_filter\_min\_pos**()

Returns the minimum filter position.

**Returns**

*int* – Returns the minimum position possible with the filter wheel.

**get\_filter\_model**()

Returns the model string from the instrument.

**Returns**

*str* – The model string of the instrument.

**get\_filter\_position**()

Returns the current filter position.

**Returns**

*int* – The current filter wheel position.

**get\_filter\_present**()

Returns if the instrument has a filter wheel.

**Returns**

*int* – Whether an integrated filter wheel is present on the filter wheel.

**get\_filter\_serial**()

Returns the serial number of the instrument.

**Returns**

*str* – The serial number of the instrument.

**static get\_filter\_preopen\_model**(*filter\_enum*)

Returns the model string of an instrument not yet opened.

Note, [\*search\\_for\\_inst\(\)\*](#) needs to be called prior to this call. In the case of multiple Filter Wheel/Spectrographs attached, it allows the user to sort, which instruments are to be opened before opening them.

**Parameters**

**filter\_enum** (*int*) – A filter enumeration value.

**Returns**

*str* – The model string of the unopened filter wheel.

**static get\_enum\_preopen\_model(*enum*)**

Returns the model number of an instrument not yet opened.

Note, [\*search\\_for\\_inst\(\)\*](#) needs to be called prior to this call.

**Parameters**

**enum** (*int*) – The enumeration for the unopened instrument.

**Returns**

*str* – A Princeton Instruments model number.

**static get\_enum\_preopen\_serial(*enum*)**

Returns the serial number of an instrument not yet opened.

Note, [\*search\\_for\\_inst\(\)\*](#) needs to be called prior to this call.

**Parameters**

**enum** (*int*) – The enumeration for the unopened instrument.

**Returns**

*str* – A Princeton Instruments serial number.

**static get\_enum\_preopen\_com(*enum*)**

Returns the COM port number of an instrument not yet opened.

Note, [\*search\\_for\\_inst\(\)\*](#) needs to be called prior to this call.

**Parameters**

**enum** (*int*) – The enumeration for the unopened instrument.

**Returns**

*int* – The COM port number.

**get\_mono\_backlash\_steps()**

Returns the number of backlash steps used when reversing the wavelength drive.

**Returns**

*int* – The number of steps the instrument backlash corrects. Not valid on older instruments.

**get\_mono\_detector\_angle()**

Returns the default detector angle of the instrument in radians.

**Returns**

*float* – The default detector angle of the instrument in radians.

**get\_mono\_diverter\_pos(*diverter\_num*)**

Returns the slit that the diverter mirror is pointing to.

**Parameters**

**diverter\_num** (*int*) – The diverter to be queried.

- Mirror Position 1 - motorized entrance diverter mirror.
- Mirror Position 2 - motorized exit diverter mirror.
- Mirror Position 3 - motorized entrance diverter on a double slave unit.
- Mirror Position 4 - motorized exit diverter on a double slave unit.

**Returns**

`int` –

The slit port that the diverter mirror is currently pointing at.

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.
- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

**get\_mono\_diverter\_pos\_str**(*diverter\_num*)

Returns a string describing the port the mirror is pointing to.

**Parameters**

**diverter\_num** (`int`) – The diverter to be queried.

- Mirror Position 1 - motorized entrance diverter mirror.
- Mirror Position 2 - motorized exit diverter mirror.
- Mirror Position 3 - motorized entrance diverter on a double slave unit.
- Mirror Position 4 - motorized exit diverter on a double slave unit.

**Returns**

`str` – A string describing which port the diverter is pointing at.

**get\_mono\_diverter\_valid**(*diverter\_num*)

Returns if a motorized diverter position is valid for an instrument.

**Parameters**

**diverter\_num** (`int`) – The diverter to be queried.

- Mirror Position 1 - motorized entrance diverter mirror.
- Mirror Position 2 - motorized exit diverter mirror.
- Mirror Position 3 - motorized entrance diverter on a double slave unit.
- Mirror Position 4 - motorized exit diverter on a double slave unit.

**Returns**

`bool` – Whether the diverter mirror is valid.

**get\_mono\_double**()

Returns if the instrument is a double monochromator.

**Returns**

`bool` – Whether the instrument is a double monochromator.

### **get\_mono\_double\_subtractive()**

Returns if a double monochromator is subtractive instead of additive.

#### **Returns**

**bool** – Whether the double monochromator is subtractive.

### **get\_mono\_double\_intermediate\_slit()**

If a monochromator is a double, return the intermediate slit position.

#### **Returns**

**slit\_pos (int)** – The intermediate slit of the double monochromator

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.
- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

### **get\_mono\_exit\_slit()**

Return the exit slit position for a monochromator.

Function works with single and double monochromators.

#### **Returns**

**slit\_pos (int)** – The intermediate slit of the double monochromator

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.
- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

### **get\_mono\_filter\_max\_pos()**

Returns the maximum filter position.

#### **Returns**

**int** – The maximum position possible with the filter wheel.

### **get\_mono\_filter\_min\_pos()**

Returns the minimum filter position.



**Returns**

`int` – The minimum position possible with the filter wheel.

**get\_mono\_filter\_position()**

Returns the current filter position.

**Returns**

`int` – The current filter wheel position.

**get\_mono\_filter\_present()**

Returns if the instrument has an integrated filter wheel.

Note, is a new option and not available on most instruments. All filter functions call this function before proceeding.

**Returns**

`bool` – Whether an integrated filter wheel is present on the monochromator.

**get\_mono\_focal\_length()**

Returns the default focal length of the instrument.

**Returns**

`float` – The focal length of the instrument in millimeters.

**get\_mono\_gear\_steps()**

Returns the number of steps in a set of sine drive gears.

**Returns**

- `int` – The number of steps per rev on the minor gear of a sine wavelength drive.
- `int` – The number of steps per rev on the major gear of a sine wavelength drive.

**get\_mono\_grating()**

Returns the current grating.

**Returns**

`int` – The current grating. This assumes the correct turret has been inserted in the instrument.

**get\_mono\_grating\_blaze(*grating*)**

Returns the blaze of a given grating.

**Parameters**

**grating** (`int`) – Which grating to request the information about. Validates the request by calling `get_mono_grating_installed()`.

**Returns**

`str` – The blaze of the grating.

**get\_mono\_grating\_density(*grating*)**

Returns the groove density (grooves per millimeter) of a given grating.

**Parameters**

**grating** (`int`) – Which grating to request the information about. Validates the request by calling `get_mono_grating_installed()`.

**Returns**

**int** – The groove density of the grating in grooves per millimeter. For a mirror, this function will return 1200 grooves per MM.

**get\_mono\_grating\_gadjust(*grating*)**

Returns the GAdjust of a grating.

**Parameters**

**grating** (**int**) – Which grating to request the information about.

**Returns**

**int** – The grating GAdjust. Note, this value is specific to a specific instrument grating combination and not transferable between instruments.

**get\_mono\_grating\_installed(*grating*)**

Returns if a grating is installed.

**Parameters**

**grating** (**int**) – Which grating we are requesting the information about.

**Returns**

**bool** – Whether the grating requested is installed.

**get\_mono\_grating\_max()**

Get the maximum grating position installed.

This is usually the number of gratings installed.

**Returns**

**int** – The number of gratings installed.

**get\_mono\_grating\_offset(*grating*)**

Returns the offset of a grating.

**Parameters**

**grating** (**int** c\_long) – The number of the grating that is to be addressed.

**Returns**

**int** – The grating offset. Note, this value is specific to a specific instrument grating combination and not transferable between instruments.

**get\_mono\_half\_angle()**

Returns the default half angle of the instrument in radians.

:param **float**: The half angle of the instrument in radians.

**get\_mono\_init\_grating()**

Returns the initial grating (on instrument reboot).

**Returns**

**int** – The power-up grating number.

**get\_mono\_init\_scan\_rate\_nm()**

Returns the initial wavelength scan rate (on instrument reboot).

**Returns**

**float** – The power-up scan rate in nm / minute.

**get\_mono\_init\_wave\_nm()**

Returns the initial wavelength (on instrument reboot).

**Returns**

`float` – The power-up wavelength in nm.

**get\_mono\_int\_led\_on()**

Checks if the interrupter LED is on.

**Returns**

`bool` – Whether the interrupter LED is on.

**get\_mono\_model()**

Returns the model number of the instrument.

**Returns**

`str` – The model number of the instrument.

**get\_mono\_motor\_int()**

Read the motor gear interrupter.

**Returns**

`bool` – Whether the motor gear was interrupted.

**get\_mono\_precision()**

Returns the nm-decimal precision of the wavelength drive.

Note, this is independent of the wavelength step resolution whose coarseness is defined by the grating being used.

**Returns**

`int` – The number of digits after the decimal point the instrument uses. Note, the true precision is limited by the density of the grating and can be much less than the instruments wavelength precision.

**get\_mono\_scan\_rate\_nm\_min()**

Return the current wavelength scan rate.

**Returns**

`float` – Scan rate in nm per minute.

**get\_mono\_serial()**

Returns the serial number of the instrument.

**Returns**

`str` – The serial number of the instrument as a string.

**get\_mono\_shutter\_open()**

Returns if the integrated shutter is open.

**Returns**

`bool` – Whether the shutter is open.

**get\_mono\_shutter\_valid()**

Returns if the instrument has an integrated shutter.

**Returns**

`bool` – Whether a shutter is present.

### **get\_mono\_sine\_drive()**

Returns if the gearing system has sine instead of a linear drive system.

#### **Returns**

**bool** – Whether the gearing is sine based verses linear gearing.

### **get\_mono\_slit\_type(*slit\_pos*)**

Returns the slit type for a given slit.

#### **Parameters**

**slit\_pos** (**int**) – The slit to be addressed.

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.
- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

#### **Returns**

**int** – The type of slit attached.

- Slit Type 0 - No slit, older instruments that do will only return a slit type of zero.
- Slit Type 1 - Manual Slit.
- Slit Type 2 - Fixed Slit Width.
- Slit Type 3 - Focal Plane adapter.
- Slit Type 4 - Continuous Motor.
- Slit Type 5 - Fiber Optic.
- Slit Type 6 - Index able Slit.

### **get\_mono\_slit\_type\_str(*slit\_pos*)**

Returns a string descriptor of a given slit.

#### **Parameters**

**slit\_pos** (**int**) – The slit to be addressed.

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.

- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

**Returns**

`str` – The description of the slit.

**get\_mono\_slit\_width(*slit\_pos*)**

Returns the slit width of a motorized slit.

**Parameters**

**slit\_pos** (`int`) – The slit to be addressed.

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.
- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

**Returns**

`int` – The width of the slit, if the slit is motorized.

**get\_mono\_slit\_width\_max(*slit\_pos*)**

Returns the maximum width of a motorized slit.

**Parameters**

**slit\_pos** (`int`) – The slit to be addressed.

- Slit Port 1 - Side entrance slit.
- Slit Port 2 - Front entrance slit.
- Slit Port 3 - Front exit slit.
- Slit Port 4 - Side exit slit.
- Slit Port 5 - Side entrance slit on a double slave unit.
- Slit Port 6 - Front entrance slit on a double slave unit.
- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

**Returns**

`int` – The maximum width of the slit, if the slit is motorized.

**get\_mono\_turret()**

Returns the current grating turret number.

**Returns**

`int` – The current turret number.

**get\_mono\_turret\_max()**

Returns the number of turrets installed.

**Returns**

**int** – The number of turrets installed.

**get\_mono\_turret\_gratings()**

Returns the number of gratings per turret.

**Returns**

**int** – The number of gratings that can be placed on a single turret. This number can be one, two or three depending on the monochromator model.

**get\_mono\_wavelength\_cutoff\_nm()**

Returns, in nm, the max wavelength achievable by the instrument using the current grating.

**Returns**

**float** – The maximum center wavelength in nanometers for the current grating. On SpectraPro's this value equals 1400 nm \* grating density / 1200 g/mm. On AM/VM products, this value is limited by the sine bar and will vary.

**get\_mono\_wavelength\_min\_nm()**

Returns, in nm, the min wavelength achievable by the instrument using the current grating.

**Returns**

**float** – The minimum center wavelength in nanometers for the current grating. On SpectraPro's it is usually -10 nm, on linear AM/VM instruments this value will vary.

**get\_mono\_wavelength\_abs\_cm()**

Returns the current center wavelength of instrument in absolute wavenumber.

**Returns**

**float** – The current wavelength of the instrument in absolute wavenumber.

**get\_mono\_wavelength\_ang()**

Returns the current center wavelength of instrument in Angstroms.

**Returns**

**float** – The current wavelength of the instrument in Angstroms.

**get\_mono\_wavelength\_ev()**

Returns the current center wavelength of instrument in electron volts.

**Returns**

**float** – The current wavelength of the instrument in electron volts.

**get\_mono\_wavelength\_micron()**

Returns the current center wavelength of instrument in microns.

**Returns**

**float** – The current wavelength of the instrument in microns.

**get\_mono\_wavelength\_nm()**

Returns the current center wavelength of instrument in nm.

**Returns**

`float` – The current wavelength of the instrument in nanometers.

**get\_mono\_wavelength\_rel\_cm(*center\_nm*)**

Returns the current center wavelength of instrument in relative wavenumber.

**Parameters**

**center\_nm** (`int`) – The wavelength in nanometers that 0 relative wavenumber is centered around.

**Returns**

`float` – The current wavelength of the instrument in relative wavenumber.

**get\_mono\_wheel\_int()**

Read the wheel gear interrupter.

**Returns**

`bool` – Whether the wheel gear was interrupted.

**get\_mono\_nm\_rev\_ratio()**

Returns the number of stepper steps per rev of the wavelength drive motor.

**Returns**

`float` – The ratio of nm per rev in a linear wavelength drive.

**static get\_mono\_preopen\_model(*mono\_enum*)**

Returns the model of an instrument not yet opened.

Note, [`search\_for\_inst\(\)`](#) needs to be called prior to this call. In the case of multiple Monochromator/Spectrographs attached, it allows the user to sort, which instruments are to be opened before opening them.

**Parameters**

**mono\_enum** (`int`) – A monochromator enumeration.

**Returns**

`str` – The model of the unopened monochromator.

**get\_ncl\_filter\_max\_pos()**

Returns the maximum filter position.

**Returns**

`int` – The maximum filter position.

**get\_ncl\_filter\_min\_pos()**

Returns the minimum filter position.

**Returns**

`int` – The minimum filter position.

**get\_ncl\_filter\_position()**

Return the current filter position.

**Returns**

`int` – The current filter position.

**get\_ncl\_filter\_present()**

Returns if the instrument has an integrated filter wheel setup.

**Returns**

`bool` – Whether a NCL filter wheel is setup and present.

**get\_ncl\_mono\_enum**(*mono\_num*)

Return the enumeration of the attached monochromator.

**Parameters**

**mono\_num** (`int`) – The readout system monochromator port being addressed.

**Returns**

`int` – The enumeration of the monochromator. Check the value returned with `valid_mono_enum()`.

**get\_ncl\_mono\_setup**(*mono\_num*)

Return if the open NCL port has a Monochromator attached.

**Parameters**

**mono\_num** (`int`) – The readout system monochromator port being addressed.

**Returns**

`bool` – Whether a monochromator is attached to the port and it is set up.

**get\_ncl\_shutter\_open**(*shutter\_num*)

Return if a NCL Readout shutter is open.

**Parameters**

**shutter\_num** (`int`) – The shutter being addressed (1 or 2 on an NCL).

**Returns**

`bool` – Whether the shutter is in the open position.

**get\_ncl\_shutter\_valid**(*shutter\_num*)

Return if a NCL Readout shutter number is valid.

**Parameters**

**shutter\_num** (`int`) – The shutter being addressed (1 or 2 on an NCL).

**Returns**

`bool` – Whether the shutter number is valid.

**get\_ncl\_ttl\_in**(*ttl\_line*)

Return if an input TTL line is being pulled on by an outside connection.

**Parameters**

**ttl\_line** (`int`) – The TTL line number being addressed.

**Returns**

`bool` – Whether *ttl\_line* is being pulled to TTL high.

**get\_ncl\_ttl\_out**(*ttl\_line*)

Return if an output TTL line is on.

**Parameters**

**ttl\_line** (`int`) – The TTL line number being addressed.

**Returns**

`bool` – Whether the output *ttl\_line* is set to TTL high.



**get\_ncl\_ttl\_valid**(*ttn\_line*)

Return if a TTL line is a valid line.

**Parameters**

**ttn\_line** (*int*) – The TTL line number being addressed.

**Returns**

*bool* – Whether *ttn\_line* is a valid line number.

**get\_num\_det**()

Return the number of single point detectors in the Readout System.

**Returns**

*int* – Number of single point detectors the readout system supports. (1 NCL-Lite, 2,3 NCL).

**static get\_num\_found\_inst\_ports**()

Returns the value last returned by *search\_for\_inst*().

Can be called multiple times.

**Returns**

*int* – The number of instruments found.

**get\_readout\_itime\_ms**()

Return the integration time used for taking a reading.

**Returns**

*int* – The integration time used for taking a reading, in milliseconds.

**get\_readout\_model**()

Returns the model string from the instrument.

:param *str*: The model string of the instrument.

**get\_readout\_serial**()

Returns the serial number from the instrument.

**Returns**

*str* – The serial number of the instrument as a string.

**static get\_readout\_preopen\_model**(*enum*)

Returns the model string of an instrument not yet opened.

Note, “meth:..*search\_for\_inst* needs to be called prior to this call. In the case of multiple Readout Systems attached, it allows the user to sort, which instruments are to be opened before opening them.

**Parameters**

**enum** (*int*) – The instrument enumeration.

**Returns**

*str* – The model string of the unopened instrument.

**set\_det\_bipolar**(*det\_num*)

Set the detector readout to bipolar (+/-).

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**set\_det\_current**(*det\_num*)

Set the detector readout type to Current.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**set\_det\_hv\_volts**(*det\_num*, *hv\_volts*)

Set the voltage value of the high voltage.

**Parameters**

- **det\_num** (*int*) – The detector to be addressed.
- **hv\_volts** (*int*) – The voltage value.

**set\_det\_hv\_off**(*det\_num*)

Set the detector high voltage to off.

**Parameters**

**det\_num** (*int* c\_long) – The detector to be addressed.

**Returns**

*bool* – Whether the high voltage has been turned off.

**set\_det\_hv\_on**(*det\_num*)

Set the detector high voltage to on.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**Returns**

*bool* – Whether the high voltage has been turned on.

**set\_det\_num\_avg\_read**(*num\_reads\_avg*)

Set the number of readings to average.

**Parameters**

**num\_reads\_avg** (*int*) – The number reads that will be required to acquire a single read.

**set\_det\_photon**(*det\_num*)

Set the detector readout type to Photon Counter.

**Parameters**

**det\_num** (*int*) – The detector to be addressed.

**set\_det\_range**(*det\_num*, *gain\_range*)

Set the detector gain-range factor.

**Parameters**

- **det\_num** (*int*) – The detector to be addressed.
- **gain\_range** (*int*) – The detector gain range.
  - Detector Range 0 - 1x
  - Detector Range 1 - 2x
  - Detector Range 2 - 4x

- Detector Range 3 - 50x
- Detector Range 4 - 200x

**set\_det\_type**(*det\_num*, *det\_type*)

Set the detector readout type.

**Parameters**

- **det\_num** (*int*) – The detector to be addressed.
- **det\_type** (*int*) – The detector readout type.
  - Detector Type 1 - Current
  - Detector Type 2 - Voltage
  - Detector Type 3 - Photon Counting

**set\_det\_unipolar**(*det\_num*)

Set the detector readout to unipolar (-).

**Parameters**

- **det\_num** (*int*) – The detector to be addressed.

**set\_det\_voltage**(*det\_num*)

Set the detector readout type to voltage.

**Parameters**

- **det\_num** (*int*) – The detector to be addressed.

**set\_filter\_position**(*position*)

Sets the current filter position.

**Parameters**

- **position** (*int*) – Position the filter is to be set to.

**set\_mono\_diverter\_pos**(*diverter\_num*, *diverter\_pos*)

Sets the port that the unit is to point to.

**Parameters**

- **diverter\_num** (*int*) – The diverter to be modified.
  - Mirror Position 1 - motorized entrance diverter mirror.
  - Mirror Position 2 - motorized exit diverter mirror.
  - Mirror Position 3 - motorized entrance diverter on a double slave unit.
  - Mirror Position 4 - motorized exit diverter on a double slave unit.
- **diverter\_pos** (*int*) – The slit port that the diverter mirror is to be set to. Not all ports are valid for all diverters.
  - Slit Port 1 - Side entrance slit.
  - Slit Port 2 - Front entrance slit. Slit Port 3 - Front exit slit.
  - Slit Port 4 - Side exit slit.
  - Slit Port 5 - Side entrance slit on a double slave unit.
  - Slit Port 6 - Front entrance slit on a double slave unit.

- Slit Port 7 - Front exit slit on a double slave unit.
- Slit Port 8 - Side exit slit on a double slave unit.

**set\_mono\_filter\_position**(*position*)

Sets the current filter position.

**Parameters**

**position** (*int*) – Position the filter is to be set to.

**set\_mono\_grating**(*grating*)

Sets the current grating.

**Parameters**

**grating** (*int*) – Set the current grating. This assumes the correct turret is inserted in the instrument and grating selected is a valid grating. This function will change to current turret in the firmware, but needs user interaction to install the correct turret.

**set\_mono\_grating\_gadjust**(*grating*, *gadjust*)

Sets the grating GAdjust.

**Parameters**

- **grating** (*int*) – The number of the grating that is to be addressed.
- **gadjust** (*int*) – The grating GAdjust. Note, this value is specific to a specific instrument-grating combination and not transferable between instruments.

**set\_mono\_grating\_offset**(*grating*, *offset*)

Sets the grating offset.

**Parameters**

- **grating** (*int*) – The number of the grating that is to be addressed.
- **offset** (*int*) – The grating offset. Note, this value is specific to a specific instrument-grating combination and not transferable between instruments.

**set\_mono\_init\_grating**(*init\_grating*)

Sets the initial grating (on instrument reboot).

**Parameters**

**init\_grating** (*int*) – The power-up grating number.

**set\_mono\_init\_scan\_rate\_nm**(*init\_scan\_rate*)

Sets the initial wavelength scan rate (on instrument reboot).

**Parameters**

**init\_scan\_rate** (*float*) – The power-up scan rate in nm / minute.

**set\_mono\_init\_wave\_nm**(*init\_wave*)

Sets the initial wavelength (on instrument reboot).

**Parameters**

**init\_wave** (*float*) – The power-up wavelength in nm.

**set\_mono\_int\_led**(*led\_state*)

Turns on and off the interrupter LED.

**Parameters**

**led\_state** (*bool*) – Indicates whether the LED should be on or off.

**set\_mono\_scan\_rate\_nm\_min**(*scan\_rate*)

Set the wavelength scan rate.

**Parameters**

**scan\_rate** (*float*) – The rate to scan the wavelength in nm/minute.

**set\_mono\_shutter\_closed**()

Sets the integrated shutter position to closed.

**Returns**

*bool* – Whether the shutter has been closed.

**set\_mono\_shutter\_open**()

Sets the integrated shutter position to open.

**Returns**

*bool* – Whether the shutter has been opened.

**set\_mono\_slit\_width**(*slit\_pos*, *slit\_width*)

Sets the slit width of a motorized slit (Types 4, 6 and 9).

**Parameters**

- **slit\_pos** (*int*) – The slit to be addressed.
  - Slit Port 1 - Side entrance slit.
  - Slit Port 2 - Front entrance slit.
  - Slit Port 3 - Front exit slit.
  - Slit Port 4 - Side exit slit.
  - Slit Port 5 - Side entrance slit on a double slave unit.
  - Slit Port 6 - Front entrance slit on a double slave unit.
  - Slit Port 7 - Front exit slit on a double slave unit.
  - Slit Port 8 - Side exit slit on a double slave unit.
- **slit\_width** (*int*) – The width to which the slit is to be set to. Note valid ranges are 10 to 3000 microns in one-micron increments for normal continuous motor slits (type 4 and 9), 10 to 12000 microns in five-micron increments for large continuous motor slits (type 4) and 25, 50, 100, 250, 500, 1000 microns for indexable slits (Type 6).

**set\_mono\_turret**(*turret*)

Sets the current grating turret.

**Parameters**

**turret** (*int*) – Set the current turret. This will change the grating to Turret number minus one times gratings per turret plus the one plus the current grating number minus one mod gratings per turret. The user must insert the correct turret into the instrument when using this function.

**set\_mono\_wavelength\_abs\_cm**(*wavelength*)

Sets the center wavelength of the instrument in absolute wavenumbers.

**Parameters**

**wavelength** (*float*) – The wavelength in absolute wavenumber that the instrument is to be moved to.

**set\_mono\_wavelength\_ang**(*wavelength*)

Sets the center wavelength of the instrument in angstroms.

**Parameters**

**wavelength** (*float*) – The wavelength in Angstroms that the instrument is to be moved to.

**set\_mono\_wavelength\_ev**(*wavelength*)

Sets the center wavelength of the instrument in Electron Volts.

**Parameters**

**wavelength** (*float*) – The wavelength in Electron Volts that the instrument is to be moved to.

**set\_mono\_wavelength\_micron**(*wavelength*)

Sets the center wavelength of the instrument in microns.

**Parameters**

**wavelength** (*float*) – The wavelength in microns that the instrument is to be moved to.

**set\_mono\_wavelength\_nm**(*wavelength*)

Sets the center wavelength of the instrument in nm.

**Parameters**

**wavelength** (*float*) – The wavelength in nm that the instrument is to be moved to.

**set\_mono\_wavelength\_rel\_cm**(*center\_nm*, *wavelength*)

Sets the center wavelength of the instrument in relative Wavenumbers.

**Parameters**

- **center\_nm** (*float*) – The wavelength in nanometers that 0 relative Wavenumber is centered around.
- **wavelength** (*float*) – The wavelength in relative Wavenumber that the instrument is to be moved to.

**set\_ncl\_filter\_position**(*position*)

Set the current filter position.

**Parameters**

**position** (*int*) – The current filter position.

**set\_ncl\_filter\_present**(*min\_filter*, *max\_filter*)

Set the instrument filter wheel to active (NCL only).

**Parameters**

- **min\_filter** (*int*) – Min Filter Position (on an NCL with the standard filter wheel 1).

- **max\_filter** (*int*) – Max Filter Position (on an NCL with the standard filter wheel 6).

**set\_ncl\_shutter\_closed**(*shutter\_num*)

Set a NCL Readout shutter to a closed state.

**Parameters**

**shutter\_num** (*int*) – The shutter being addressed (1 or 2 on an NCL).

**set\_ncl\_shutter\_open**(*shutter\_num*)

Set a NCL Readout shutter to a closed state.

**Parameters**

**shutter\_num** (*int*) – The shutter being addressed (1 or 2 on an NCL).

**set\_ncl\_ttl\_out\_off**(*ttn\_line*)

Turn a TTL line off.

**Parameters**

**ttn\_line** (*int*) – The TTL line number being addressed.

**set\_ncl\_ttl\_out\_on**(*ttn\_line*)

Turn a TTL line on.

**Parameters**

**ttn\_line** (*int*) – The TTL line number being addressed.

**set\_readout\_itime\_ms**(*itime\_ms*)

Set the integration time used for taking a reading.

**Parameters**

**itime\_ms** (*int*) – The integration time in milliseconds to be used when reading out a detector.

**static init**(*path*='ARC\_Instrument\_x64.dll')

Initialize the SDK.

**Parameters**

**path** (*str*) – The path to the SDK.

**static is\_initiated**()

Check if the *init()* method was called.

**Returns**

*bool* – Whether the *init()* method was called.

**static error\_to\_english**(*error\_code*)

Convert an error code into a message.

**Parameters**

**error\_code** (*int*) – The error code.

**Returns**

*str* – The error message.

## msl.equipment.resources.raicol package

Resources for equipment from Raicol Crystals.

### Submodules

## msl.equipment.resources.raicol.raicol\_tec module

Control a TEC (Peltier-based) oven from Raicol Crystals.

**class** msl.equipment.resources.raicol.raicol\_tec.RaicolTEC(*record*)

Bases: *ConnectionSerial*

Control a TEC (Peltier-based) oven from Raicol Crystals.

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**get\_setpoint()**

Get the setpoint temperature.

#### Returns

*float* – The setpoint temperature, in Celsius.

**off()**

Turn the TEC off.

**on()**

Turn the TEC on.

**set\_setpoint**(*temperature*)

Set the setpoint temperature.

#### Parameters

**temperature** (*float*) – The setpoint temperature, in Celsius. Must be in the range [20.1, 60.0].

**temperature()**

Returns the current temperature of the oven.

The temperature is measured by a PT1000-Platinum resistor temperature sensor that is located near the crystal in the metallic mount.

#### Returns

*float* – The temperature of the oven, in Celsius.



## msl.equipment.resources.thorlabs package

Resources for equipment 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 Thorlabs Kinesis v1.14.10

## msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor module

This module provides all the functionality required to control a **Benchtop Stepper Motor** including:

- BSC101
- BSC102
- BSC103
- BSC201
- BSC202
- BSC203

**class** msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor()

Bases: [MotionControl](#)

A wrapper around Thorlabs.MotionControl.Benchtop.StepperMotor.dll.

The *properties* for a BenchtopStepperMotor connection supports the following key-value pairs in the *Connections Database*:

```
'device_name': str, the device name found in ThorlabsDefaultSettings.xml,
↪ [default: None]
```

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

#### Parameters

**record** ([EquipmentRecord](#)) – A record from an *Equipment-Register Database*.

**can\_home**(*channel*)

Can the device perform a *home()*?

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*bool* – *True* if the device can perform a home.

**can\_move\_without\_homing\_first**(*channel*)

Does the device need to be *home()*'d before a move can be performed?

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*bool* – *True* if the device does not need to be *home()*'d before a move can be commanded.

**check\_connection**()

Check connection.

**Returns**

*bool* – *True* if the USB is listed by the FTDI controller.

**clear\_message\_queue**(*channel*)

Clears the device message queue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**close**()

Disconnect and close the device.

**Raises**

*ThorlabsError* – If not successful.

**disable\_channel**(*channel*)

Disable the channel so that the motor can be moved by hand.

When disabled, power is removed from the motor and it can be freely moved.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**enable\_channel**(*channel*)

Enable channel for computer control.

When enabled, power is applied to the motor so it is fixed in position.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

***ThorlabsError*** – If not successful.

**enable\_last\_msg\_timer**(*channel*, *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**

- **channel** (*int*) – The channel number (1 to n).
- **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**(*channel*)

Get the backlash distance setting (used to control hysteresis).

See *get\_real\_value\_from\_device\_unit()* for converting from a `DeviceUnit` to a `RealValue`.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The backlash distance in `DeviceUnits` (see manual).

**get\_bow\_index**(*channel*)

Gets the stepper motor bow index.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The bow index.

**get\_calibration\_file**(*channel*)

Get the calibration file for this motor.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*str* – The filename of the calibration file.

**Raises**

***ThorlabsError*** – If not successful.

**get\_device\_unit\_from\_real\_value**(*channel*, *real\_value*, *unit\_type*)

Converts a real-world value to a device value.

Either *load\_settings()*, *load\_named\_settings()* or *set\_motor\_params\_ext()* must be called before calling this function, otherwise the returned value will always be 0.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).

- **real\_value** (*float*) – The real-world value.
- **unit\_type** (*enums.UnitType*) – The unit of the real-world value.

**Returns**

*int* – The device value.

**Raises**

*ThorlabsError* – If not successful.

**get\_digital\_outputs**(*channel*)

Gets the digital output bits.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – Bit mask of states of the 4 digital output pins.

**get\_encoder\_counter**(*channel*)

Get the Encoder Counter.

For devices that have an encoder, the current encoder position can be read.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – Encoder count of encoder units.

**get\_firmware\_version**(*channel*)

Gets the version number of the device firmware.

**Returns**

*str* – The firmware version.

**get\_hardware\_info**(*channel*)

Gets the hardware information from the device.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*structs.TLI\_HardwareInformation* – The hardware information.

**Raises**

*ThorlabsError* – If not successful.

**get\_hardware\_info\_block**(*channel*)

Gets the hardware information in a block.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*structs.TLI\_HardwareInformation* – The hardware information.

**Raises**

*ThorlabsError* – If not successful.

**get\_homing\_params\_block(channel)**

Get the homing parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*structs.MOT\_HomingParameters* – The homing parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_homing\_velocity(channel)**

Gets the homing velocity.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The homing velocity in DeviceUnits (see manual).

**get\_input\_voltage(channel)**

Gets the analogue input voltage reading.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The input voltage 0-32768 corresponding to 0-5V.

**get\_jog\_mode(channel)**

Gets the jog mode.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

- *enums.MOT\_JogModes* – The jog mode.
- *enums.MOT\_StopModes* – The stop mode.

**Raises**

*ThorlabsError* – If not successful.

**get\_jog\_params\_block(channel)**

Get the jog parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*structs.MOT\_JogParameters* – The jog parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_jog\_step\_size(*channel*)**

Gets the distance to move when jogging.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a `DeviceUnit` to a `RealValue`.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

`int` – The step size in `DeviceUnits` (see manual).

**get\_jog\_vel\_params(*channel*)**

Gets the jog velocity parameters.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a `DeviceUnit` to a `RealValue`.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

- `int` – The maximum velocity in `DeviceUnits` (see manual).
- `int` – The acceleration in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_joystick\_params(*channel*)**

Gets the joystick parameters.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

[`structs.MOT\_JoystickParameters`](#) – The joystick parameters.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_limit\_switch\_params(*channel*)**

Gets the limit switch parameters.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**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**

`ThorlabsError` – If not successful.

**get\_limit\_switch\_params\_block**(*channel*)

Get the limit switch parameters.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

`structs.MOT_LimitSwitchParameters` – The limit switch parameters.

**Raises**

`ThorlabsError` – If not successful.

**get\_motor\_params**(*channel*)

Gets the motor stage parameters.

Deprecated: calls `get_motor_params_ext()`

**get\_motor\_params\_ext**(*channel*)

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`.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

- `float` – The steps per revolution.
- `float` – The gear box ratio.
- `float` – The pitch.

**Raises**

`ThorlabsError` – If not successful.

**get\_motor\_travel\_limits**(*channel*)

Gets the motor stage min and max position.

These define the range of travel for the stage.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

- `float` – The minimum position in `RealWorldUnits` [millimeters or degrees].
- `float` – The maximum position in `RealWorldUnits` [millimeters or degrees].

**Raises**

`ThorlabsError` – If not successful.

**get\_motor\_travel\_mode(channel)**

Get the motor travel mode.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*enums.MOT\_TravelModes* – The travel mode.

**get\_motor\_velocity\_limits(channel)**

Gets the motor stage maximum velocity and acceleration.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

- *float* – The maximum velocity in RealWorldUnits [millimeters or degrees].
- *float* – The maximum acceleration in RealWorldUnits [millimeters or degrees].

**Raises**

*ThorlabsError* – If not successful.

**get\_move\_absolute\_position(channel)**

Gets the move absolute position.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The move absolute position in DeviceUnits (see manual).

**get\_move\_relative\_distance(channel)**

Gets the move relative distance.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The move relative position in DeviceUnits (see manual).

**get\_next\_message(channel)**

Get the next Message Queue item, if it is available. See *messages*.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

- *int* – The message type.
- *int* – The message ID.



- `int` – The message data.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_num\_channels()**

Gets the number of channels in the device.

**Returns**

`int` – The number of channels.

**get\_number\_positions(channel)**

Get the 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.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

`int` – The number of positions.

**get\_pid\_loop\_encoder\_coeff(channel)**

Gets the Encoder PID loop encoder coefficient.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

`float` – The Encoder PID loop encoder coefficient.

**get\_pid\_loop\_encoder\_params(channel)**

Gets the Encoder PID loop parameters.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

[`structs.MOT\_PIDLoopEncoderParams`](#) – The parameters used to define the Encoder PID Loop.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_position(channel)**

Get the current position.

The current position is the last recorded position. The current position is updated either by the polling mechanism or by calling [`request\_position\(\)`](#).

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a `DeviceUnit` to a `RealValue`.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

**index** (`int`) – The current position in `DeviceUnits` (see manual).

**get\_position\_counter(channel)**

Get the position counter.

The position counter is identical to the position parameter. The position counter is set to zero when homing is complete. The position counter can also be set using [`set\_position\_counter\(\)`](#) if homing is not to be performed.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The position counter in DeviceUnits (see manual).

**get\_power\_params(channel)**

Gets the power parameters for the stepper motor.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

[`structs.MOT\_PowerParameters`](#) – The power parameters.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_rack\_digital\_outputs()**

Gets the rack digital output bits.

**Returns**

*int* – Bit mask of states of the 4 digital output pins.

**get\_rack\_status\_bits()**

Gets the Rack status bits.

**Returns**

*int* – The status bits including 4 with one per electronic input pin.

**get\_real\_value\_from\_device\_unit(channel, device\_value, unit\_type)**

Converts a device value to a real-world value.

Either [`load\_settings\(\)`](#), [`load\_named\_settings\(\)`](#) or [`set\_motor\_params\_ext\(\)`](#) must be called before calling this function, otherwise the returned value will always be 0.0.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **device\_value** (*int*) – The device value.
- **unit\_type** ([`enums.UnitType`](#)) – The unit of the device value.

**Returns**

*float* – The real-world value.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_soft\_limit\_mode(channel)**

Gets the software limits mode.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**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(channel)**

Gets the Stepper Motor maximum stage position.

See [\*get\\_real\\_value\\_from\\_device\\_unit\(\)\*](#) for converting from a DeviceUnit to a RealValue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The maximum position in DeviceUnits (see manual).

**get\_stage\_axis\_min\_pos(channel)**

Gets the Stepper Motor minimum stage position.

See [\*get\\_real\\_value\\_from\\_device\\_unit\(\)\*](#) for converting from a DeviceUnit to a RealValue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The minimum position in DeviceUnits (see manual).

**get\_status\_bits(channel)**

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 the polling function, [\*start\\_polling\(\)\*](#).

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The status bits from the device.

**get\_trigger\_switches(channel)**

Gets the trigger switch parameter.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

`int` – Trigger mask where:

- Bit 0 - Input trigger enabled.
- Bit 1 - Output trigger enabled.
- Bit 2 - Output Passthrough mode enabled where Output Trigger mirrors Input Trigger.
- Bit 3 - Output trigger high when moving.
- Bit 4 - Performs relative move when input trigger goes high.
- Bit 5 - Performs absolute move when input trigger goes high.
- Bit 6 - Performs home when input trigger goes high.
- Bit 7 - Output triggers when motor moved by software command.

**get\_vel\_params(channel)**

Gets the move velocity parameters.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a `DeviceUnit` to a `RealValue`.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

- **max\_velocity** (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_vel\_params\_block(channel)**

Get the move velocity parameters.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

[`structs.MOT\_VelocityParameters`](#) – The velocity parameters.

**Raises**

[`ThorlabsError`](#) – If not successful.

**has\_last\_msg\_timer\_overrun(channel)**

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.

**Parameters**

**channel** (`int`) – The channel number (1 to n).

**Returns**

**bool** – **True** if last message timer has elapsed, **False** if monitoring is not enabled or if time of last message received is less than `lastMsgTimeout`.

**home(channel)**

Home the device.

Homing the device will set the device to a known state and determine the home position.

**Parameters**

**channel** (**int**) – The channel number (1 to n).

**Raises**

**ThorlabsError** – If not successful.

**identify(channel)**

Sends a command to the device to make it identify itself.

**Parameters**

**channel** (**int**) – The channel number (1 to n).

**Raises**

**ThorlabsError** – If not successful.

**is\_calibration\_active(channel)**

Is a calibration file active for this motor?

**Parameters**

**channel** (**int**) – The channel number (1 to n).

**Returns**

**bool** – Whether a calibration file is active.

**is\_channel\_valid(channel)**

Verifies that the specified channel is valid.

**Parameters**

**channel** (**int**) – The requested channel number (1 to n).

**Returns**

**bool** – Whether the channel is valid.

**load\_settings(channel)**

Update device with stored settings.

The settings are read from `ThorlabsDefaultSettings.xml`, which gets created when the Kinesis software is installed.

**Parameters**

**channel** (**int**) – The channel number (1 to n).

**Raises**

**ThorlabsError** – If not successful.

**load\_named\_settings(channel, settings\_name)**

Update device with named settings.

**Parameters**

- **channel** (**int**) – The channel number (1 to n).

- **settings\_name** (*str*) – The name of the device to load the settings for. Examples for the value of *setting\_name* can be found in *ThorlabsDefaultSettings.xml*, which gets created when the Kinesis software is installed.

**Raises**

*ThorlabsError* – If not successful.

**max\_channel\_count()**

Gets the number of channels available to this device.

This function returns the number of available bays, not the number of bays filled.

**Returns**

*int* – The number of channels available on this device.

**message\_queue\_size(channel)**

Gets the size of the message queue.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The number of messages in the queue.

**move\_absolute(channel)**

Moves the device to the position defined in the *set\_move\_absolute\_position()* command.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**move\_at\_velocity(channel, direction)**

Start moving at the current velocity in the specified direction.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **direction** (*enums.MOT\_TravelDirection*) – The required direction of travel as a *enums.MOT\_TravelDirection* enum value or member name.

**Raises**

*ThorlabsError* – If not successful.

**move\_jog(channel, jog\_direction)**

Perform a jog.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **jog\_direction** (*enums.MOT\_TravelDirection*) – The jog direction as a *enums.MOT\_TravelDirection* enum value or member name.

**Raises**

***ThorlabsError*** – If not successful.

**move\_relative**(*channel*, *displacement*)

Move the motor by a relative amount.

See [\*get\\_device\\_unit\\_from\\_real\\_value\(\)\*](#) for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **displacement** (*int*) – Signed displacement in `DeviceUnits` (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**move\_relative\_distance**(*channel*)

Moves the device by a relative distance defined by [\*set\\_move\\_relative\\_distance\(\)\*](#).

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

***ThorlabsError*** – If not successful.

**move\_to\_position**(*channel*, *index*)

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 `RealValue` to a `DeviceUnit`.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **index** (*int*) – The position in `DeviceUnits` (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**needs\_homing**(*channel*)

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**

***ThorlabsError*** – If not successful.

**persist\_settings(channel)**

Persist device settings to device.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**polling\_duration(channel)**

Gets the polling loop duration.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

*int* – The time between polls in milliseconds or 0 if polling is not active.

**register\_message\_callback(channel, callback)**

Registers a callback on the message queue.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **callback** (*MotionControlCallback*) – A function to be called whenever messages are received.

**Raises**

*ThorlabsError* – If not successful.

**request\_backlash(channel)**

Requests the backlash.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_bow\_index(channel)**

Requests the stepper motor bow index.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_digital\_outputs(channel)**

Requests the digital output bits.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_encoder\_counter(channel)**

Requests the encoder counter.

For devices that have an encoder, the current encoder position can be read.



**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_homing\_params**(*channel*)

Requests the homing parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_input\_voltage**(*channel*)

Requests the analogue input voltage reading.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_jog\_params**(*channel*)

Requests the jog parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_joystick\_params**(*channel*)

Requests the joystick parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_limit\_switch\_params**(*channel*)

Requests the limit switch parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_absolute\_position**(*channel*)

Requests the position of next absolute move.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_relative\_distance**(*channel*)

Requests the relative move distance.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_pid\_loop\_encoder\_params**(*channel*)

Requests the Encoder PID loop parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_position**(*channel*)

Requests the current position.

This needs to be called to get the device to send its current position. Note, this is called automatically if Polling is enabled for the device using *start\_polling()*.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_power\_params**(*channel*)

Requests the power parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_rack\_digital\_outputs**()

Requests the rack digital output bits.

**Raises**

*ThorlabsError* – If not successful.

**request\_rack\_status\_bits**()

Requests the Rack status bits be downloaded.

**Raises**

*ThorlabsError* – If not successful.

**request\_settings**(*channel*)

Requests that all settings are downloaded from the device.

This function requests that the device upload all its settings to the DLL.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_status\_bits**(*channel*)

Request the status bits which identify the current motor state.

This needs to be called to get the device to send its current status bits. Note, this is called automatically if *Polling* is enabled for the device using *start\_polling()*.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_trigger\_switches**(*channel*)

Requests the trigger switch parameter.

**Warning:** This function is currently not in the DLL, as of v1.14.8, but it is in the header file.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**request\_vel\_params**(*channel*)

Requests the velocity parameters.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**reset\_rotation\_modes**(*channel*)

Reset the rotation modes for a rotational device.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**resume\_move\_messages**(*channel*)

Resume suspended move messages.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

*ThorlabsError* – If not successful.

**set\_backlash**(*channel*, *distance*)

Sets the backlash distance (used to control hysteresis).

See [`get\_device\_unit\_from\_real\_value\(\)`](#) for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **distance** (`int`) – The backlash distance in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_bow\_index**(*channel*, *bow\_index*)

Sets the stepper motor bow index.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **bow\_index** (`int`) – The bow index.

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_calibration\_file**(*channel*, *path*, *enabled*)

Set the calibration file for this motor.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **path** (`str`) – The path to a calibration file to load.
- **enabled** (`bool`) – `True` to enable, `False` to disable.

**Raises**

[`OSError`](#) – If the *path* does not exist.

**set\_digital\_outputs**(*channel*, *outputs\_bits*)

Sets the digital output bits.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **outputs\_bits** (`int`) – Bit mask to set states of the 4 digital output pins.

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_direction**(*channel*, *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**

- **channel** (`int`) – The channel number (1 to n).

- **reverse** (*bool*) – If *True* then directions will be swapped on these moves.

**Raises**

*ThorlabsError* – If not successful.

**set\_encoder\_counter**(*channel, count*)

Set the Encoder Counter values.

Setting the encoder counter to zero, effectively defines a home position on the encoder strip. Setting this value does not move the device.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **count** (*int*) – The encoder count in encoder units.

**Raises**

*ThorlabsError* – If not successful.

**set\_homing\_params\_block**(*channel, direction, limit, velocity, offset*)

Set the homing parameters.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **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**

*ThorlabsError* – If not successful.

**set\_homing\_velocity**(*channel, velocity*)

Sets the homing velocity.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **velocity** (*int*) – The homing velocity in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_jog\_mode**(*channel, mode, stop\_mode*)

Sets the jog mode.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **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**

*ThorlabsError* – If not successful.

**set\_jog\_params\_block**(*channel, jog\_params*)

Set the jog parameters.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **jog\_params** (*structs.MOT\_JogParameters*) – The jog parameters.

**Raises**

*ThorlabsError* – If not successful.

**set\_jog\_step\_size**(*channel, step\_size*)

Sets the distance to move on jogging.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **step\_size** (*int*) – The step size in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_jog\_vel\_params**(*channel, max\_velocity, acceleration*)

Sets jog velocity parameters.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **max\_velocity** (*int*) – The maximum velocity in *DeviceUnits* (see manual).
- **acceleration** (*int*) – The acceleration in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_joystick\_params**(*channel, joystick\_params*)

Sets the joystick parameters.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **joystick\_params** (`structs.MOT_JoystickParameters`) – The joystick parameters.

**Raises**

**ThorlabsError** – If not successful.

**set\_limit\_switch\_params**(*channel*, *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 `RealValue` to a `DeviceUnit`.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **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**

**ThorlabsError** – If not successful.

**set\_limit\_switch\_params\_block**(*channel*, *params*)

Set the limit switch parameters.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).
- **params** (`structs.MOT_LimitSwitchParameters`) – The new limit switch parameters.

**Raises**

**ThorlabsError** – If not successful.

**set\_limits\_software\_approach\_policy**(*channel*, *policy*)

Sets the software limits policy.

**Parameters**

- **channel** (`int`) – The channel number (1 to n).

- **policy** (*enums.MOT\_LimitsSoftwareApproachPolicy*) – The soft limit mode as a *enums.MOT\_LimitsSoftwareApproachPolicy* enum value or member name.

**set\_motor\_params**(*channel, steps\_per\_rev, gear\_box\_ratio, pitch*)

Sets the motor stage parameters.

Deprecated: calls *set\_motor\_params\_ext()*

**set\_motor\_params\_ext**(*channel, 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  $\text{steps\_per\_rev} * \text{gear\_box\_ratio} / \text{pitch}$ .

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

#### Parameters

- **channel** (*int*) – The channel number (1 to n).
- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

#### Raises

*ThorlabsError* – If not successful.

**set\_motor\_travel\_limits**(*channel, 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 DeviceUnit to a RealValue.

#### Parameters

- **channel** (*int*) – The channel number (1 to n).
- **min\_position** (*float*) – The minimum position in RealWorldUnits [millimeters or degrees].
- **max\_position** (*float*) – The maximum position in RealWorldUnits [millimeters or degrees].

#### Raises

*ThorlabsError* – If not successful.

**set\_motor\_travel\_mode**(*channel, travel\_mode*)

Set the motor travel mode.

#### Parameters

- **channel** (*int*) – The channel number (1 to n).
- **travel\_mode** (*enums.MOT\_TravelModes*) – The travel mode as a *enums.MOT\_TravelModes* enum value or member name.



**Raises**

***ThorlabsError*** – If not successful.

**set\_motor\_velocity\_limits**(*channel*, *max\_velocity*, *max\_acceleration*)

Sets the motor stage maximum velocity and acceleration.

See [\*get\\_real\\_value\\_from\\_device\\_unit\(\)\*](#) for converting from a DeviceUnit to a RealValue.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **max\_velocity** (*float*) – The maximum velocity in RealWorldUnits [millimeters or degrees].
- **max\_acceleration** (*float*) – The maximum acceleration in RealWorldUnits [millimeters or degrees].

**Raises**

***ThorlabsError*** – If not successful.

**set\_move\_absolute\_position**(*channel*, *position*)

Sets the move absolute position.

See [\*get\\_device\\_unit\\_from\\_real\\_value\(\)\*](#) for converting from a RealValue to a DeviceUnit.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **position** (*int*) – The absolute position in DeviceUnits (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**set\_move\_relative\_distance**(*channel*, *distance*)

Sets the move relative distance.

See [\*get\\_device\\_unit\\_from\\_real\\_value\(\)\*](#) for converting from a RealValue to a DeviceUnit.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **distance** (*int*) – The relative position in DeviceUnits (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**set\_pid\_loop\_encoder\_coeff**(*channel*, *coeff*)

Sets the Encoder PID loop encoder coefficient.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **coeff** (*float*) – The Encoder PID loop encoder coefficient. Set to 0.0 to disable the encoder or if no encoder is present otherwise the positive encoder coefficient.

Raises

[\*ThorlabsError\*](#) – If not successful.

**set\_pid\_loop\_encoder\_params**(*channel, mode, prop\_gain, int\_gain, diff\_gain, limit, tol*)

Sets the Encoder PID loop parameters.

Parameters

- **channel** (*int*) – The channel number (1 to n).
- **mode** (*enums.MOT\_PID\_LoopMode*) – The Encoder PID loop mode as a *enums.MOT\_PID\_LoopMode* enum value or member name.
- **prop\_gain** (*int*) – The Encoder PID Loop proportional gain. Range 0 to  $2^{24}$ .
- **int\_gain** (*int*) – The Encoder PID Loop integral gain. Range 0 to  $2^{24}$ .
- **diff\_gain** (*int*) – The Encoder PID Loop differential gain. Range 0 to  $2^{24}$ .
- **limit** (*int*) – The Encoder PID Loop output limit. Range 0 to  $2^{15}$ .
- **tol** (*int*) – The Encoder PID Loop tolerance. Range 0 to  $2^{15}$ .

Raises

[\*ThorlabsError\*](#) – If not successful.

**set\_position\_counter**(*channel, 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 *RealValue* to a *DeviceUnit*.

Parameters

- **channel** (*int*) – The channel number (1 to n).
- **count** (*int*) – The position counter in *DeviceUnits* (see manual).

Raises

[\*ThorlabsError\*](#) – If not successful.

**set\_power\_params**(*channel, rest, move*)

Sets the power parameters for the stepper motor.

Parameters

- **channel** (*int*) – The channel number (1 to n).
- **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

[\*ThorlabsError\*](#) – If not successful.

**set\_rack\_digital\_outputs**(*outputs\_bits*)

Sets the rack digital output bits.

**Parameters**

**outputs\_bits** (*int*) – Bit mask to set states of the 4 digital output pins.

**Raises**

*ThorlabsError* – If not successful.

**set\_rotation\_modes**(*channel, mode, direction*)

Set the rotation modes for a rotational device.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **mode** (*enums.MOT\_MovementModes*) – The travel mode as a *enums.MOT\_MovementModes* enum value or member name.
- **direction** (*enums.MOT\_MovementDirections*) – The travel mode as a *enums.MOT\_MovementDirections* enum value or member name.

**Raises**

*ThorlabsError* – If not successful.

**set\_stage\_axis\_limits**(*channel, min\_position, max\_position*)

Sets the stage axis position limits.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **min\_position** (*int*) – The minimum position in *DeviceUnits* (see manual).
- **max\_position** (*int*) – The maximum position in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_trigger\_switches**(*channel, indicator\_bits*)

Sets the trigger switch bits.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **indicator\_bits** (*int*) – Sets the 8 bits indicating action on trigger input and events to trigger electronic output.
  - Bit 0 - Input trigger enabled.
  - Bit 1 - Output trigger enabled.
  - Bit 2 - Output pass-through mode enabled where Output Trigger mirrors Input Trigger.
  - Bit 3 - Output trigger high when moving.

- Bit 4 - Performs relative move when input trigger goes high.
- Bit 5 - Performs absolute move when input trigger goes high.
- Bit 6 - Performs home when input trigger goes high.
- Bit 7 - Output triggers when motor moved by software command.

**Raises**

***ThorlabsError*** – If not successful.

**set\_vel\_params**(*channel*, *max\_velocity*, *acceleration*)

Sets the move velocity parameters.

See [\*get\\_device\\_unit\\_from\\_real\\_value\(\)\*](#) for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **max\_velocity** (*int*) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (*int*) – The acceleration in `DeviceUnits` (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**set\_vel\_params\_block**(*channel*, *min\_velocity*, *max\_velocity*, *acceleration*)

Set the move velocity parameters.

See [\*get\\_device\\_unit\\_from\\_real\\_value\(\)\*](#) for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **min\_velocity** (*int*) – The minimum velocity in `DeviceUnits` (see manual).
- **max\_velocity** (*int*) – The maximum velocity in `DeviceUnits` (see manual)..
- **acceleration** (*int*) – The acceleration in `DeviceUnits` (see manual)..

**Raises**

***ThorlabsError*** – If not successful.

**start\_polling**(*channel*, *milliseconds*)

Starts the internal polling loop.

This function continuously requests position and status messages.

**Parameters**

- **channel** (*int*) – The channel number (1 to n).
- **milliseconds** (*int*) – The polling rate, in milliseconds.

**Raises**

***ThorlabsError*** – If not successful.

**stop\_immediate**(*channel*)

Stop the current move immediately (with the risk of losing track of the position).

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

***ThorlabsError*** – If not successful.

**stop\_polling**(*channel*)

Stops the internal polling loop.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**stop\_profiled**(*channel*)

Stop the current move using the current velocity profile.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

***ThorlabsError*** – If not successful.

**suspend\_move\_messages**(*channel*)

Suspend automatic messages at ends of moves.

Useful to speed up part of real-time system with lots of short moves.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Raises**

***ThorlabsError*** – If not successful.

**time\_since\_last\_msg\_received**(*channel*)

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.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

- *int* – The time, in milliseconds, since the last message was received.
- *bool* – *True* if monitoring is enabled otherwise *False*.

**uses\_pid\_loop\_encoding**(*channel*)

Determines if we can use PID loop encoding.

This is true if the stage supports PID Loop Encoding. Requires *get\_pid\_loop\_encoder\_coeff()* to have a positive non-zero coefficient, see also *set\_pid\_loop\_encoder\_coeff()*.

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**wait\_for\_message**(*channel*)

Wait for next Message Queue item if it is available. See [messages](#).

**Parameters**

**channel** (*int*) – The channel number (1 to n).

**Returns**

- *int* – The message type.
- *int* – The message ID.
- *int* – The message data.

**Raises**

[ThorlabsError](#) – If not successful.

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

A callback to register for a *MotionControl* message queue.

```
from msl.equipment import Config
from msl.equipment.resources.thorlabs import MotionControlCallback

@MotionControlCallback
def msg_callback():
    print('MotionControlCallback: ', flipper.convert_message(*flipper.get_
    ↪next_message()))

# The "example2.xml" configuration file contains the following element:
# <equipment alias="filter_flipper" manufacturer="Thorlabs" model="MFF101/M"/>

db = Config('config.xml').database()

flipper = db.equipment['filter_flipper'].connect()
flipper.register_message_callback(msg_callback)

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

**msl.equipment.resources.thorlabs.kinesis.callbacks.MotionControlCallback**

A callback to register for a *MotionControl* message queue.

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

Enums defined in Thorlabs Kinesis v1.14.10

```
class msl.equipment.resources.thorlabs.kinesis.enums.FT_Status(value, names=None,
                                                                *, module=None,
                                                                qualname=None,
                                                                type=None, start=1,
                                                                boundary=None)
```

Bases: [IntEnum](#)

```
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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
MOT_NotMotor = 0
```

```
MOT_DCMotor = 1
```

```
MOT_StepperMotor = 2
```

```
MOT_BrushlessMotor = 3
```

```
MOT_CustomMotor = 100
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_TravelModes(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
MOT_TravelModeUndefined = 0
```

```
MOT_Linear = 1
```

**MOT\_Rotational = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_TravelDirection(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**MOT\_TravelDirectionDisabled = 0**

**MOT\_Forwards = 1**

**MOT\_Reverse = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_DirectionSense(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**MOT\_Normal = 0**

**MOT\_Backwards = 1**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_HomeLimitSwitchDirection(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**MOT\_LimitSwitchDirectionUndefined = 0**

**MOT\_ReverseLimitSwitch = 1**



**MOT\_ForwardLimitSwitch = 4**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_JogModes(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**MOT\_JogModeUndefined = 0**

**MOT\_Continuous = 1**

**MOT\_SingleStep = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_StopModes(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**MOT\_StopModeUndefined = 0**

**MOT\_Immediate = 1**

**MOT\_Profiled = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_ButtonModes(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**MOT\_ButtonModeUndefined = 0**

**MOT\_JogMode = 1**

**MOT\_Preset = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_VelocityProfileModes(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`MOT_Trapezoidal = 0`

`MOT_SCurve = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_LimitSwitchModes(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

`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(value,
                                     names=None,
                                     *,
                                     module=None,
                                     qualname=None,
                                     type=None,
                                     start=1,
                                     boundary=None)
```

Bases: `IntEnum`

`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_LimitsSoftwareApproachPolicy(value,
                                               names=
                                               *,
                                               module=
                                               qual=
                                               name=
                                               type=N
                                               start=1
                                               bound=
                                               ary=N
```

Bases: `IntEnum`

`DisallowIllegalMoves = 0`

`AllowPartialMoves = 1`

`AllowAllMoves = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_CurrentLoopPhases(value,
                                                                            names=None,
                                                                            *,
                                                                            module=None,
                                                                            qual-
                                                                            name=None,
                                                                            type=None,
                                                                            start=1,
                                                                            bound-
                                                                            ary=None)
```

Bases: `IntEnum`

**MOT\_PhaseA** = 0

**MOT\_PhaseB** = 1

**MOT\_PhaseAB** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_MovementModes(value,
                                                                           names=None,
                                                                           *, module-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**LinearRange** = 0

**RotationalUnlimited** = 1

**RotationalWrapping** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_MovementDirections(value,
                                                                                names=None,
                                                                                *,
                                                                                mod-
                                                                                ule=None,
                                                                                qual-
                                                                                name=None,
                                                                                type=None,
                                                                                start=1,
                                                                                bound-
                                                                                ary=None)
```

Bases: `IntEnum`

**Quickest** = 0

**Forwards** = 1

**Reverse = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.MOT_PID_LoopMode(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**MOT\_PIDLoopModeDisabled = 0**

**MOT\_PIDOpenLoopMode = 1**

**MOT\_PIDClosedLoopMode = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_SignalState(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**NT\_BadSignal = 0**

**NT\_GoodSignal = 1**

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_Mode(value, names=None, *,
                                                                module=None,
                                                                qualname=None,
                                                                type=None, start=1,
                                                                boundary=None)
```

Bases: `IntEnum`

**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(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=
                                                                    None)

Bases: IntEnum

NT_ControlModeUndefined = 0

NT_OpenLoop = 1

NT_ClosedLoop = 2

NT_OpenLoopSmoothed = 3

NT_ClosedLoopSmoothed = 4

class msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSource(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=
                                                                    None)

Bases: IntEnum

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(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)

Bases: IntEnum
```

```
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(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
NT_OddAndEven = 1
```

```
NT_Odd = 2
```

```
NT_Even = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_UnderOrOver(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
NT_InRange = 1
```

```
NT_UnderRange = 2
```

```
NT_OverRange = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_CircleDiameterMode(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

```
Bases: IntEnum
```

```
NT_ParameterCircleMode = 1
```

```
NT_AbsPowerCircleMode = 2
```

```
NT_LUTCircleMode = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_CircleAdjustment(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

```
Bases: IntEnum
```

```
NT_LinearCircleAdjustment = 1
```

```
NT_LogCircleAdjustment = 2
```

```
NT_SquareCircleAdjustment = 3
```

```
NT_CubeCircleAdjustment = 4
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_TIARangeMode(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```



Bases: `IntEnum`

`NT_TTIARangeModeUndefined = 0`

`NT_AutoRangeAtSelected = 1`

`NT_ManualRangeAtSelected = 2`

`NT_ManualRangeAtParameter = 3`

`NT_AutoRangeAtParameter = 4`

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_LowPassFrequency(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`NT_VoltageRangeUndefined = 0`

`NT_VoltageRange_5v = 1`

`NT_VoltageRange_10v = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_OutputVoltageRoute(value,
                                                                           names=None,
                                                                           *,
                                                                           module=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

`NT_SMAOnly = 1`

`NT_HubOrSMA = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_PowerInputUnits(value,
                                                                           names=None,
                                                                           *, module=
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

`NT_Amps = 0`

`NT_Watts = 1`

`NT_Db = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_SMA_Units(value,
                                                                      names=None, *,
                                                                      module=None,
                                                                      qual-
                                                                      name=None,
                                                                      type=None,
                                                                      start=1, bound-
                                                                      ary=None)
```

Bases: `IntEnum`

`NT_Voltage = 0`

`NT_FullRange = 1`

`NT_UserDefined = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.BNT_CurrentLimit(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
NT_CurrentLimit_100mA = 0
```

```
NT_CurrentLimit_250mA = 1
```

```
NT_CurrentLimit_500mA = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.BNT_OutputLowPassFilter(value,
                                                                    names=None,
                                                                    *,
                                                                    module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
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(value,
                                                                    names=None,
                                                                    *,
                                                                    module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
NT_FeedbackSignalDC = 0
```

**NT\_FeedbackSignalAC = 65535**

```
class msl.equipment.resources.thorlabs.kinesis.enums.BNT_BNCTriggerModes(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**NT\_BNCModeTrigger = 0**

**NT\_BNCModeLVOut = 65535**

```
class msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**PZ\_Undefined = 0**

**PZ\_OpenLoop = 1**

**PZ\_CloseLoop = 2**

**PZ\_OpenLoopSmooth = 3**

**PZ\_CloseLoopSmooth = 4**

```
class msl.equipment.resources.thorlabs.kinesis.enums.PZ_InputSourceFlags(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**PZ\_SoftwareOnly = 0**

```
PZ_ExternalSignal = 1
```

```
PZ_Potentiometer = 2
```

```
PZ_All = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.PZ_OutputLUTModes(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
DerivFilterOn = 1
```

```
DerivFilterOff = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_NotchFilterState(value,
                                                                           names=None,
                                                                           *,
                                                                           module=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**NotchFilterOn** = 1

**NotchFilterOff** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_NotchFilterChannel(value,
                                                                              names=None,
                                                                              *,
                                                                              module-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**NotchFilter1** = 1

**NotchFilter2** = 2

**NotchFilterBoth** = 3

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOControlMode(value,
                                                                           names=None,
                                                                           *, module-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**SWOnly** = 0

**ExtBNC** = 1

**Joystick** = 2

**JoystickBnc** = 3

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOOutputMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**HV** = 1

**PosRaw** = 2

**PosCorrected** = 3

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOOutputBandwidth(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**OP\_Unfiltered** = 1

**OP\_200Hz** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_IOFeedbackSourceDefinition(value,
                                                                                      names=None,
                                                                                      *,
                                                                                      mod-
                                                                                      ule=None,
                                                                                      qual-
                                                                                      name=None,
                                                                                      type=None,
                                                                                      start=1,
                                                                                      bound-
                                                                                      ary=None)
```

Bases: `IntEnum`

**StrainGauge** = 1

**Capacitive** = 2

**Optical = 3**

```
class msl.equipment.resources.thorlabs.kinesis.enums.PPC_DisplayIntensity(value,
                                                                           names=None,
                                                                           *,
                                                                           module=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

**Bright = 1**

**Dim = 2**

**Off = 3**

```
class msl.equipment.resources.thorlabs.kinesis.enums.FF_Positions(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**FF\_PositionError = 0**

**Position1 = 1**

**Position2 = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.FF_IOModes(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**FF\_ToggleOnPositiveEdge = 1**

**FF\_SetPositionOnPositiveEdge = 2**

**FF\_OutputHighAtSetPosition = 4**

**FF\_OutputHighWhenMoving = 8**



```
class msl.equipment.resources.thorlabs.kinesis.enums.FF_SignalModes(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

`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_WheelDirectionSense(value,
                                                                                names=None,
                                                                                *,
                                                                                module=None,
                                                                                qualname=None,
                                                                                type=None,
                                                                                start=1,
                                                                                boundary=None)
```

Bases: `IntEnum`

`KMOT_WM_Positive = 1`

`KMOT_WM_Negative = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.KMOT_WheelMode(value,
                                                                       names=None,
                                                                       *, module=None,
                                                                       qualname=None,
                                                                       type=None,
                                                                       start=1,
                                                                       boundary=None)
```

Bases: `IntEnum`

`KMOT_WM_Velocity = 1`

```
KMOT_WM_Jog = 2
```

```
KMOT_WM_MoveAbsolute = 3
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KMOT_TriggerPortMode(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

```
Bases: IntEnum
```

```
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(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

```
Bases: IntEnum
```

```
KMOT_TrigPolarityHigh = 1
```

```
KMOT_TrigPolarityLow = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_Channels(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**Channel1** = 1

**Channel2** = 2

**Channel3** = 3

**Channel4** = 4

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_JogMode(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**JogContinuous** = 1

**JogStep** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_TravelDirection(value,
                                                                              names=None,
                                                                              *, mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**Forward** = 1

**Reverse** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_FBSignalMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)

Bases: IntEnum

FB_LimitSwitch = 1

FB_Encoder = 2

class msl.equipment.resources.thorlabs.kinesis.enums.KIM_LimitSwitchModes(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)

Bases: IntEnum

Ignore = 1

SwitchMakes = 2

SwitchBreaks = 3

SwitchMakes_HomeOnly = 4

SwitchBreaks_HomeOnly = 5

class msl.equipment.resources.thorlabs.kinesis.enums.KIM_DirectionSense(value,
                                                                            names=None,
                                                                            *, mod-
                                                                            ule=None,
                                                                            qual-
                                                                            name=None,
                                                                            type=None,
                                                                            start=1,
                                                                            bound-
                                                                            ary=None)

Bases: IntEnum

Dir_Disabled = 0
```

```
Dir_Forward = 1
```

```
Dir_Reverse = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_TrigModes(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

```
Trig_Disabled = 0
```

```
Trig_In_GPI = 1
```

```
Trig_InRelativeMove = 2
```

```
Trig_InAbsoluteMove = 3
```

```
Trig_InResetCount = 4
```

```
Trig_Out_GP0 = 10
```

```
Trig_Out_InMotion = 11
```

```
Trig_Out_AtMaxVelocity = 12
```

```
Trig_Out_PosStepFwd = 13
```

```
Trig_Out_PosStepRev = 14
```

```
Trig_Out_PosStepBoth = 15
```

```
Trig_Out_AtFwdLimit = 16
```

```
Trig_Out_AtRevLimit = 17
```

```
Trig_Out_AtEitherLimit = 18
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_TrigPolarities(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

```
Bases: IntEnum
```

Trig\_High = 1

Trig\_Low = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.KIM_JoysticModes(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

JS\_Velocity = 1

JS\_Jog = 2

JS\_GotoPosition = 3

```
class msl.equipment.resources.thorlabs.kinesis.enums.ChannelEnableModes(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

NONE = 0

Channel1Only = 1

Channel2Only = 2

Channel3Only = 3

Channel4Only = 4

Channels1and2 = 5

Channels3and4 = 6

```
class msl.equipment.resources.thorlabs.kinesis.enums.LD_InputSourceFlags(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           qualname=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

`LD_SoftwareOnly = 1`

`LD_ExternalSignal = 2`

`LD_Potentiometer = 4`

```
class msl.equipment.resources.thorlabs.kinesis.enums.LD_DisplayUnits(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           qualname=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

`LD_ILim = 1`

`LD_ILD = 2`

`LD_IPD = 3`

`LD_PLD = 4`

```
class msl.equipment.resources.thorlabs.kinesis.enums.LD_TIA_RANGES(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           qualname=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

`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(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

```
Bases: IntEnum
```

```
LD_CathodeGrounded = 1
```

```
LD_AnodeGrounded = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KLD_TriggerMode(value,
                                                                        names=None,
                                                                        *, module=
                                                                        None, qual-
                                                                        name=None,
                                                                        type=None,
                                                                        start=1,
                                                                        bound-
                                                                        ary=None)
```

```
Bases: IntEnum
```

```
KLD_Disabled = 0
```

```
KLD_Input = 1
```

```
KLD_Output = 10
```

```
KLD_LaserOn = 11
```

```
KLD_InterlockEnabled = 12
```

```
KLD_SetPointChange = 13
```

```
KLD_HighStability = 14
```

```
KLD_LowStability = 15
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KLD_TrigPolarity(value,
                                                                           names=None,
                                                                           *, module=
                                                                           None, qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```



Bases: `IntEnum`

`KLD_TrigPol_High = 1`

`KLD_TrigPol_Low = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.LS_InputSourceFlags(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

`LS_SoftwareOnly = 0`

`LS_ExternalSignal = 1`

`LS_Potentiometer = 4`

```
class msl.equipment.resources.thorlabs.kinesis.enums.KLS_OpMode(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

`KLS_ConstantPower = 0`

`KLS_ConstantCurrent = 1`

```
class msl.equipment.resources.thorlabs.kinesis.enums.KLS_TriggerMode(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

`KLS_Disabled = 0`

`KLS_Input = 1`

```
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(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
KLS_TrigPol_High = 1
KLS_TrigPol_Low = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackSource(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
NT_FeedbackSourceUndefined = 0
NT_TIA = 1
NT_IO1_5v = 4
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_TIARange(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
KNA_TTIARange1_5nA = 3
KNA_TTIARange2_16_6nA = 4
KNA_TTIARange3_50nA = 5
KNA_TTIARange4_166nA = 6
KNA_TTIARange5_500nA = 7
KNA_TTIARange6_1_66uA = 8
KNA_TTIARange7_5uA = 9
KNA_TTIARange8_16_6uA = 10
KNA_TTIARange9_50uA = 11
KNA_TTIARange10_166uA = 12
KNA_TTIARange11_500uA = 13
KNA_TTIARange12_1_66mA = 14
KNA_TTIARange13_5mA = 15
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_LowVoltageRange(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: [IntEnum](#)

```
KNA_VoltageRange_10v = 2
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_LowOutputVoltageRoute(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: [IntEnum](#)

```
KNA_IO1Only = 1
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_HighVoltageRange(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
KNA_Default_Range = 0
KNA_VoltageRange_CH1_75v = 0
KNA_VoltageRange_CH1_150v = 1
KNA_VoltageRange_CH2_75v = 0
KNA_VoltageRange_CH2_150v = 16
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_HighOutputVoltageRoute(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

```
KNA_Default_Route = 0
KNA_ExtIn_PIN = 0
KNA_ExtIn_IO1 = 1
KNA_ExtOut_Dis = 0
KNA_ExtOut_IO2 = 16
KNA_EnableInputBoost = 256
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.NT_IO1_Units(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`NT_Voltage = 0`

`NT_FullRange = 1`

`NT_UserDefined = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_WheelAdjustRate(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`KNA_WM_Low = 0`

`KNA_WM_Medium = 1`

`KNA_WM_High = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_TriggerPortMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`KNA_TrigDisabled = 0`

`KNA_TrigIn_GPI = 1`

`KNA_TrigIn_VoltageStepUp = 2`

`KNA_TrigIn_VoltageStepDown = 3`

`KNA_TrigOut_GPO = 10`

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_TriggerPortPolarity(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**KNA\_TrigPolarityHigh = 1**

**KNA\_TrigPolarityLow = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_Channels(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**KNA\_ChannelUndefined = 0**

**KNA\_Channel1 = 1**

**KNA\_Channel2 = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackModeTypes(value,
                                                                              names=None,
                                                                              *,
                                                                              mod-
                                                                              ule=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**PZ\_ControlModeUndefined = 0**

**PZ\_OpenLoop = 1**

**PZ\_CloseLoop = 2**

**PZ\_OpenLoopSmooth = 3**

**PZ\_CloseLoopSmooth = 4**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KPZ_WheelDirectionSense(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**KPZ\_WM\_Positive = 1**

**KPZ\_WM\_Negative = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KPZ_WheelMode(value,
                                                                      names=None,
                                                                      *,
                                                                      module=None,
                                                                      qual-
                                                                      name=None,
                                                                      type=None,
                                                                      start=1,
                                                                      bound-
                                                                      ary=None)
```

Bases: `IntEnum`

**KPZ\_WM\_MoveAtVoltage = 1**

**KPZ\_WM\_JogVoltage = 2**

**KPZ\_WM\_SetVoltage = 3**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KPZ_WheelChangeRate(value,
                                                                              names=None,
                                                                              *, module=
                                                                              None, qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**KPZ\_WM\_High = 1**

**KPZ\_WM\_Medium = 2**

**KPZ\_WM\_Low = 3**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KPZ_TriggerPortMode(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           qualname=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

**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(value,
                                                                                names=None,
                                                                                *,
                                                                                module=None,
                                                                                qualname=None,
                                                                                type=None,
                                                                                start=1,
                                                                                boundary=None)
```

Bases: `IntEnum`

**KPZ\_TrigPolarityHigh = 1**

**KPZ\_TrigPolarityLow = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.HubAnalogueModes(value,
                                                                           names=None,
                                                                           *, module=None,
                                                                           qualname=None,
                                                                           type=None,
                                                                           start=1,
                                                                           boundary=None)
```

Bases: `IntEnum`

**AnalogueCh1 = 1**



**AnalogueCh2** = 2

**ExtSignalSMA** = 3

```
class msl.equipment.resources.thorlabs.kinesis.enums.QD_OperatingMode(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**QD\_ModeUndefined** = 0

**QD\_Monitor** = 1

**QD\_OpenLoop** = 2

**QD\_ClosedLoop** = 3

**QD\_AutoOpenClosedLoop** = 4

```
class msl.equipment.resources.thorlabs.kinesis.enums.QD_LowVoltageRoute(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**QD\_RouteUndefined** = 0

**QD\_SMAOnly** = 1

**QD\_HubAndSMA** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.QD_OpenLoopHoldValues(value,
                                                                    names=None,
                                                                    *,
                                                                    mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`QD_HoldOnZero = 1`

`QD_HoldOnLastValue = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.QD_FilterEnable(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`QD_Undefined = 0`

`QD_Enabled = 1`

`QD_Disabled = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.QD_KPA_TrigModes(value,
                                                                    names=None,
                                                                    *, mod-
                                                                    ule=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

`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(value,
                                                                           names=None,
                                                                           *,
                                                                           mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

`GD_Trig_High = 1`

`GD_Trig_Low = 2`

```
class msl.equipment.resources.thorlabs.kinesis.enums.SC_OperatingModes(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

`SC_Manual = 1`

`SC_Single = 2`

`SC_Auto = 3`

`SC_Triggered = 4`

```
class msl.equipment.resources.thorlabs.kinesis.enums.SC_OperatingStates(value,
                                                                           names=None,
                                                                           *, mod-
                                                                           ule=None,
                                                                           qual-
                                                                           name=None,
                                                                           type=None,
                                                                           start=1,
                                                                           bound-
                                                                           ary=None)
```

Bases: `IntEnum`

`SC_Unknown = 0`

`SC_Active = 1`

**SC\_Inactive = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.SC_SolenoidStates(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**SC\_SolenoidOpen = 1**

**SC\_SolenoidClosed = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KSC_TriggerPortMode(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**KSC\_TrigDisabled = 0**

**KSC\_TrigIn\_GPI = 1**

**KSC\_TrigOut\_GPO = 10**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KSC_TriggerPortPolarity(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**KSC\_TrigPolarityHigh = 1**

**KSC\_TrigPolarityLow = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.KST_Stages(value,
                                                                names=None, *,
                                                                module=None,
                                                                qualname=None,
                                                                type=None,
                                                                start=1,
                                                                boundary=None)
```

Bases: `IntEnum`

**ZST6 = 32**

**ZST13 = 33**

**ZST25 = 34**

**ZST206 = 48**

**ZST213 = 49**

**ZST225 = 50**

**ZFS206 = 64**

**ZFS213 = 65**

**ZFS225 = 66**

**DRV013\_25MM = 80**

**DRV014\_50MM = 81**

**NR360 = 112**

**PLS\_X25MM = 114**

**PLS\_X25MM\_HiRes = 115**

**FW103 = 117**

```
class msl.equipment.resources.thorlabs.kinesis.enums.TSG_Hub_Analogue_Modes(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**TSG\_HubChannel1 = 1**

**TSG\_HubChannel2 = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.TSG_Display_Modes(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)

Bases: IntEnum

TSG_Undefined = 0

TSG_Position = 1

TSG_Voltage = 2

TSG_Force = 3

class msl.equipment.resources.thorlabs.kinesis.enums.KSG_TriggerPortMode(value,
                                                                    names=None,
                                                                    *, module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)

Bases: IntEnum

KSG_TrigDisabled = 0

KSG_TrigIn_GPI = 1

KSG_TrigOut_GPO = 10

KSG_TrigOut_LessThanLowerLimit = 11

KSG_TrigOut_MoreThanLowerLimit = 12

KSG_TrigOut_LessThanUpperLimit = 13

KSG_TrigOut_MoreThanUpperLimit = 14

KSG_TrigOut_BetweenLimits = 15

KSG_TrigOut_OutsideLimits = 16
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.KSG_TriggerPortPolarity(value,
                                                                              names=None,
                                                                              *,
                                                                              module=None,
                                                                              qual-
                                                                              name=None,
                                                                              type=None,
                                                                              start=1,
                                                                              bound-
                                                                              ary=None)
```

Bases: `IntEnum`

**KSG\_TrigPolarityHigh = 1**

**KSG\_TrigPolarityLow = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.TIM_Channels(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1, bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**Channel1 = 1**

**Channel2 = 2**

**Channel3 = 3**

**Channel4 = 4**

```
class msl.equipment.resources.thorlabs.kinesis.enums.TIM_JogMode(value,
                                                                    names=None, *,
                                                                    module=None,
                                                                    qualname=None,
                                                                    type=None,
                                                                    start=1,
                                                                    boundary=None)
```

Bases: `IntEnum`

**JogContinuous = 1**

**JogStep = 2**

```
class msl.equipment.resources.thorlabs.kinesis.enums.TIM_ButtonsMode(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**Jog** = 1

**Position** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.TIM_Direction(value,
                                                                    names=None,
                                                                    *,
                                                                    module=None,
                                                                    qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**Forward** = 1

**Reverse** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.LS_DisplayUnits(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**LS\_mAmps** = 1

**LS\_mWatts** = 2

**LS\_mDb** = 3



```
class msl.equipment.resources.thorlabs.kinesis.enums.TST_Stages(value,
                                                                names=None, *,
                                                                module=None,
                                                                qualname=None,
                                                                type=None,
                                                                start=1,
                                                                boundary=None)

Bases: IntEnum

ZST6 = 32

ZST13 = 33

ZST25 = 34

ZST206 = 48

ZST213 = 49

ZST225 = 50

ZFS206 = 64

ZFS213 = 65

ZFS225 = 66

TBD1 = 96

TBD2 = 97

TBD3 = 98

TBD4 = 99

NR360 = 112

MVS025 = 113

PLS_X25MM = 114

PLS_X25MM_HiRes = 115

FW103 = 117

NEWZFS06 = 10006

NEWZFS13 = 10013

NEWZFS25 = 10025

NEWZST06 = 11006

NEWZST13 = 11013

NEWZST25 = 12025
```

```
class msl.equipment.resources.thorlabs.kinesis.enums.TC_SensorTypes(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**TC\_Transducer** = 0

**TC\_TH20kOhm** = 1

**TC\_TH200kOhm** = 2

```
class msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes(value,
                                                                    names=None,
                                                                    *, module=
                                                                    None, qual-
                                                                    name=None,
                                                                    type=None,
                                                                    start=1,
                                                                    bound-
                                                                    ary=None)
```

Bases: `IntEnum`

**TC\_ActualTemperature** = 0

**TC\_TargetTemperature** = 1

**TC\_TempDifference** = 2

**TC\_Current** = 3

```
class msl.equipment.resources.thorlabs.kinesis.enums.UnitType(value, names=None,
                                                                *, module=None,
                                                                qualname=None,
                                                                type=None, start=1,
                                                                boundary=None)
```

Bases: `IntEnum`

**DISTANCE** = 0

**VELOCITY** = 1

**ACCELERATION** = 2

**msl.equipment.resources.thorlabs.kinesis.errors module**

Device and Low Level Error Codes defined in Thorlabs Kinesis v1.14.10

**msl.equipment.resources.thorlabs.kinesis.filter\_flipper module**

This module provides all the functionality required to control a Filter Flipper (MFF101, MFF102).

**class** msl.equipment.resources.thorlabs.kinesis.filter\_flipper.**FilterFlipper**(*record*)

Bases: *MotionControl*

A wrapper around Thorlabs.MotionControl.FilterFlipper.dll.

The *properties* for a FilterFlipper connection supports the following key-value pairs in the *Connections Database*:

```
'device_name': str, the device name found in ThorlabsDefaultSettings.xml.
↪ [default: None]
```

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**load\_named\_settings(settings\_name)**

Update device with named settings.

**Parameters**

**settings\_name** (*str*) – The name of the device to load the settings for. Examples for the value of *setting\_name* can be found in *ThorlabsDefaultSettings.xml*, which gets created when the Kinesis software is installed.

**Raises**

*ThorlabsError* – If not successful.

**persist\_settings()**

Persist the devices current settings.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**move\_to\_position(position)**

Move the device to the specified position (index).

**Parameters**

**position** (*int*) – The required position. Must be 1 or 2.

**Raises**

*ThorlabsError* – If not successful.

**get\_position()**

Get the current position.

**Returns**

*int* – The position, 1 or 2 (can be 0 during a move).

**get\_io\_settings()**

Gets the I/O settings from filter flipper.

**Returns**

*FF\_IOSettings* – The Filter Flipper I/O settings.

**Raises**

*ThorlabsError* – If not successful.

**request\_io\_settings()**

Requests the I/O settings from the filter flipper.

**Raises**

*ThorlabsError* – If not successful.

**set\_io\_settings**(*transit\_time=500*, *oper1=FF\_IOModes.FF\_ToggleOnPositiveEdge*,  
*sig1=FF\_SignalModes.FF\_InputButton*, *pw1=200*,  
*oper2=FF\_IOModes.FF\_ToggleOnPositiveEdge*,  
*sig2=FF\_SignalModes.FF\_OutputLevel*, *pw2=200*)

Sets the settings on filter flipper.

**Parameters**

- **transit\_time** (*int*, optional) – Time taken to get from one position to other in milliseconds.
- **oper1** (*FF\_IOModes*, optional) – I/O 1 Operating Mode.
- **sig1** (*FF\_SignalModes*, optional) – I/O 1 Signal Mode.
- **pw1** (*int*, optional) – Digital I/O 1 pulse width in milliseconds.
- **oper2** (*FF\_IOModes*, optional) – I/O 2 Operating Mode.
- **sig2** (*FF\_SignalModes*, optional) – I/O 2 Signal Mode.
- **pw2** (*int*, optional) – Digital I/O 2 pulse width in milliseconds.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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*, optional) – 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 it's settings to the DLL.

**Raises**

*ThorlabsError* – If not successful.

**clear\_message\_queue()**

Clears the device message queue.

**register\_message\_callback(callback)**

Registers a callback on the message queue.

**Parameters**

**callback** (*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**

*ThorlabsError* – 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**

*ThorlabsError* – 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, MLJ150)
- Cage Rotator (K10CR1)

**class** `msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperM`

Bases: *MotionControl*

A wrapper around `Thorlabs.MotionControl.IntegratedStepperMotors.dll`.

The *properties* for an `IntegratedStepperMotors` connection supports the following key-value pairs in the *Connections Database*:

```
'device_name': str, the device name found in ThorlabsDefaultSettings.xml.  
↪ [default: None]
```

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A 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**

[`ThorlabsError`](#) – 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**

[`ThorlabsError`](#) – 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 `DeviceUnit` to a `RealValue`.

**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 `DeviceUnit` to a `RealValue`.

**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**

*ThorlabsError* – If not successful.

**get\_button\_params\_block()**

Get the button parameters.

**Returns**

*structs.MOT\_ButtonParameters* – The button parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_calibration\_file()**

Get the calibration file for this motor.

**Returns**

`str` – The filename of the calibration file.

**Raises**

*ThorlabsError* – 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()*, *load\_named\_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** (*enums.UnitType*) – The unit of the real-world value.

**Returns**

`int` – The device value.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**get\_hardware\_info\_block()**

Gets the hardware information in a block.

**Returns**

*structs.TLI\_HardwareInformation* – The hardware information.

**Raises**

*ThorlabsError* – If not successful.

**get\_homing\_params\_block()**

Get the homing parameters.

**Returns**

*structs.MOT\_HomingParameters* – The homing parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_homing\_velocity()**

Gets the homing velocity.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**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**

*ThorlabsError* – If not successful.

**get\_jog\_params\_block()**

Get the jog parameters.

**Returns**

*structs.MOT\_JogParameters* – The jog parameters.

**Raises**

*ThorlabsError* – 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 DeviceUnit to a RealValue.

**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 `DeviceUnit` to a `RealValue`.

**Returns**

- `int` – The maximum velocity in `DeviceUnits` (see manual).
- `int` – The acceleration in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – 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 `DeviceUnit` to a `RealValue`.

**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**

[`ThorlabsError`](#) – If not successful.

**get\_limit\_switch\_params\_block()**

Get the limit switch parameters.

**Returns**

[`structs.MOT\_LimitSwitchParameters`](#) – The limit switch parameters.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_motor\_params()**

Gets the motor stage parameters.

Deprecated: calls [`get\_motor\_params\_ext\(\)`](#)

**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**

- `class`float`` – The steps per revolution.
- `float` – The gear box ratio.
- `float` – The pitch.

**Raises**

***ThorlabsError*** – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – If not successful.

**get\_move\_absolute\_position()**

Gets the move absolute position.

See *`get_real_value_from_device_unit()`* for converting from a `DeviceUnit` to a `RealValue`.

**Returns**

`int` – The move 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 DeviceUnit to a RealValue.

**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**

[ThorlabsError](#) – If not successful.

**get\_number\_positions()**

Get the 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 DeviceUnit to a RealValue.

**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 DeviceUnit to a RealValue.

**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 DeviceUnit to a RealValue.

**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**

*ThorlabsError* – If not successful.

**get\_potentiometer\_params\_block()**

Get the potentiometer parameters.

**Returns**

*structs.MOT\_PotentiometerSteps* – The potentiometer parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_power\_params()**

Gets the power parameters for the stepper motor.

**Returns**

*structs.MOT\_PowerParameters* – The power parameters.

**Raises**

*ThorlabsError* – 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()*, *load\_named\_settings()* or *set\_motor\_params\_ext()* must be called before calling this function, otherwise the returned value will always be 0.

**Parameters**

- **device\_value** (`int`) – The device value.
- **unit\_type** (*enums.UnitType*) – The unit of the device value.

**Returns**

`float` – The real-world value.

**Raises**

*ThorlabsError* – 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 DeviceUnit to a RealValue.

**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 DeviceUnit to a RealValue.

**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 function, [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 DeviceUnit to a RealValue.

**Returns**

- **max\_velocity** ([int](#)) – The maximum velocity in DeviceUnits (see manual).
- **acceleration** ([int](#)) – The acceleration in DeviceUnits (see manual).

**Raises**

[ThorlabsError](#) – If not successful.

**get\_vel\_params\_block()**

Get the move velocity parameters.

**Returns**

[structs.MOT\\_VelocityParameters](#) – The velocity parameters.

**Raises**

[ThorlabsError](#) – 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, `False` if monitoring is not enabled or if time of last message received is less than `lastMsgTimeout`.

**home()**

Home the device.

Homing the device will set the device to a known state and determine the home position.

**Raises**

*ThorlabsError* – If not successful.

**identify()**

Sends a command to the device to make it identify itself.

**Raises**

*ThorlabsError* – If not successful.

**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**

*ThorlabsError* – If not successful.

**load\_named\_settings(settings\_name)**

Update device with named settings.

**Parameters**

**settings\_name** (`str`) – The name of the device to load the settings for. Examples for the value of *setting\_name* can be found in *ThorlabsDefaultSettings.xml*, which gets created when the Kinesis software is installed.

**Raises**

*ThorlabsError* – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – If not successful.

**move\_relative(*displacement*)**

Move the motor by a relative amount.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

**displacement** (*int*) – Signed displacement in *DeviceUnits* (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**move\_relative\_distance()**

Moves the device by a relative distance defined by *set\_move\_relative\_distance()*.

**Raises**

***ThorlabsError*** – If not successful.

**move\_to\_position(*index*)**

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 *RealValue* to a *DeviceUnit*.

**Parameters**

**index** (*int*) – The position in *DeviceUnits* (see manual).

**Raises**

***ThorlabsError*** – 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**

*ThorlabsError* – If not successful.

**persist\_settings()**

Persist the devices current settings.

**Raises**

*ThorlabsError* – 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** (*MotionControlCallback*) – A function to be called whenever messages are received.

**request\_backlash()**

Requests the backlash.

**Raises**

*ThorlabsError* – If not successful.

**request\_bow\_index()**

Requests the stepper motor bow index.

**Raises**

*ThorlabsError* – If not successful.

**request\_button\_params()**

Requests the LTS button parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_homing\_params()**

Requests the homing parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_jog\_params()**

Requests the jog parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_limit\_switch\_params()**

Requests the limit switch parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_absolute\_position()**

Requests the position of next absolute move.

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_relative\_distance()**

Requests the relative move distance.

**Raises**

*ThorlabsError* – If not successful.

**request\_position()**

Requests the current position.

This needs to be called to get the device to send its current position. Note, this is called automatically if Polling is enabled for the device using *start\_polling()*.

**Raises**

*ThorlabsError* – If not successful.

**request\_potentiometer\_params()**

Requests the potentiometer parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_power\_params()**

Requests the power parameters.

**Raises**

*ThorlabsError* – If not successful.

**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**

*ThorlabsError* – 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**

*ThorlabsError* – 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 its current status bits. Note, this is called automatically if Polling is enabled for the device using [start\\_polling\(\)](#).

**Raises**

[ThorlabsError](#) – If not successful.

**request\_trigger\_switches()**

Requests the trigger switch bits.

**Raises**

[ThorlabsError](#) – If not successful.

**request\_vel\_params()**

Requests the velocity parameters.

**Raises**

[ThorlabsError](#) – If not successful.

**reset\_rotation\_modes()**

Reset the rotation modes for a rotational device.

**Raises**

[ThorlabsError](#) – If not successful.

**reset\_stage\_to\_defaults()**

Reset the stage settings to defaults.

**Raises**

[ThorlabsError](#) – 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 RealValue to a DeviceUnit.

**Parameters**

**distance** ([int](#)) – The backlash distance in DeviceUnits (see manual).

**Raises**

[ThorlabsError](#) – If not successful.

**set\_bow\_index(*bow\_index*)**

Sets the stepper motor bow index.

**Parameters**

**bow\_index** ([int](#)) – The bow index.

**Raises**

[ThorlabsError](#) – 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 RealValue to a DeviceUnit.

**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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*OSError* – 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.

**Raises**

*ThorlabsError* – If not successful.

**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**

*ThorlabsError* – If not successful.

**set\_homing\_velocity**(*velocity*)

Sets the homing velocity.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

**velocity** (*int*) – The homing velocity in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**set\_jog\_params\_block**(*jog\_params*)

Set the jog parameters.

**Parameters**

**jog\_params** (*structs.MOT\_JogParameters*) – The jog parameters.

**Raises**

- *ThorlabsError* – 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 `RealValue` to a `DeviceUnit`.

**Parameters**

**step\_size** (`int`) – The step size in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – 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 `RealValue` to a `DeviceUnit`.

**Parameters**

- **max\_velocity** (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – 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**

[`ThorlabsError`](#) – 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 `RealValue` to a `DeviceUnit`.

**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**

*ThorlabsError* – If not successful.

**set\_limit\_switch\_params\_block**(*params*)

Set the limit switch parameters.

**Parameters**

**params** (*structs.MOT\_LimitSwitchParameters*) – The new limit switch parameters.

**Raises**

- *ThorlabsError* – If not successful.
- *TypeError* – If the data type of *joystick\_params* is not *structs.MOT\_JoystickParameters*

**set\_limits\_software\_approach\_policy**(*policy*)

Sets the software limits policy.

**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  $\text{steps\_per\_rev} * \text{gear\_box\_ratio} / \text{pitch}$ .

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises**

*ThorlabsError* – 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  $\text{steps\_per\_rev} * \text{gear\_box\_ratio} / \text{pitch}$ .

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises**

*ThorlabsError* – 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 DeviceUnit to a RealValue.

**Parameters**

- **min\_position** (*float*) – The minimum position in RealWorldUnits [millimeters or degrees].
- **max\_position** (*float*) – The maximum position in RealWorldUnits [millimeters or degrees].

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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 DeviceUnit to a RealValue.

**Parameters**

- **max\_velocity** (*float*) – The maximum velocity in RealWorldUnits [millimeters or degrees].
- **max\_acceleration** (*float*) – The maximum acceleration in RealWorldUnits [millimeters or degrees].

**Raises**

*ThorlabsError* – If not successful.

**set\_move\_absolute\_position**(*position*)

Sets the move absolute position.

See *get\_device\_unit\_from\_real\_value()* for converting from a RealValue to a DeviceUnit.

**Parameters**

**position** (*int*) – The absolute position in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_move\_relative\_distance**(*distance*)

Sets the move relative distance.

See *get\_device\_unit\_from\_real\_value()* for converting from a RealValue to a DeviceUnit.

**Parameters**

**distance** (*int*) – The relative position in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – 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 RealValue to a DeviceUnit.

**Parameters**

**count** (*int*) – The position counter in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – 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 RealValue to a DeviceUnit.

**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**

*ThorlabsError* – If not successful.

**set\_potentiometer\_params\_block**(*params*)

Set the potentiometer parameters.

**Parameters**

**params** (*structs.MOT\_PotentiometerSteps*) – The potentiometer parameters.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**set\_rotation\_modes**(*mode, direction*)

Set the rotation modes for a rotational device.

**Parameters**

- **mode** (*enums.MOT\_MovementModes*) – The travel mode as a *enums.MOT\_MovementModes* enum value or member name.
- **direction** (*enums.MOT\_MovementDirections*) – The travel mode as a *enums.MOT\_MovementDirections* enum value or member name.

**Raises**

*ThorlabsError* – 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 *RealValue* to a *DeviceUnit*.

**Parameters**

- **min\_position** (*int*) – The minimum position in *DeviceUnits* (see manual).
- **max\_position** (*int*) – The maximum position in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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 *RealValue* to a *DeviceUnit*.

**Parameters**

- **max\_velocity** (*int*) – The maximum velocity in DeviceUnits (see manual).
- **acceleration** (*int*) – The acceleration in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_vel\_params\_block**(*min\_velocity*, *max\_velocity*, *acceleration*)

Set the move velocity parameters.

See *get\_device\_unit\_from\_real\_value()* for converting from a RealValue to a DeviceUnit.

**Parameters**

- **min\_velocity** (*int*) – The minimum velocity in DeviceUnits (see manual)..
- **max\_velocity** (*int*) – The maximum velocity in DeviceUnits (see manual)..
- **acceleration** (*int*) – The acceleration in DeviceUnits (see manual)..

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**stop\_immediate**()

Stop the current move immediately (with the risk of losing track of the position).

**Raises**

*ThorlabsError* – If not successful.

**stop\_polling**()

Stops the internal polling loop.

**stop\_profiled**()

Stop the current move using the current velocity profile.

**Raises**

*ThorlabsError* – 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.
- `bool` – `True` if monitoring is enabled otherwise `False`.

**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**

[ThorlabsError](#) – If not successful.

**msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo module**

This module provides all the functionality required to control a KCube DC Servo (KDC101).

**class** `msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo(record)`

Bases: [MotionControl](#)

A wrapper around `Thorlabs.MotionControl.KCube.DCServo.dll`.

The [properties](#) for a KCubeDCServo connection supports the following key-value pairs in the [Connections Database](#):

`'device_name': str`, the device name found **in** `ThorlabsDefaultSettings.xml`.  
↪ `[default: None]`

Do not instantiate this class directly. Use the [connect\(\)](#) method to connect to the equipment.

**Parameters**

**record** ([EquipmentRecord](#)) – A record from an [Equipment-Register Database](#).

**can\_device\_lock\_front\_panel()**

Determine if the device front panel can be locked.

**Returns**

`bool` – `True` if the front panel of the device can be locked, `False` if not.

**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**

**ThorlabsError** – 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**

**ThorlabsError** – 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 `DeviceUnit` to a `RealValue`.

**Returns**

**int** – The backlash distance in `DeviceUnits` (see manual).

**get\_dcpid\_params()**

Get the DC PID parameters for DC motors used in an algorithm involving calculus.

**Returns**

**structs.MOT\_DC\_PIDParameters** – The DC PID parameters.

**Raises**

**ThorlabsError** – 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()`, `load_named_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** (*enums.UnitType*) – The unit of the real-world value.

**Returns**

*int* – The device value.

**Raises**

*ThorlabsError* – If not successful.

**get\_digital\_outputs**()

Gets the digital output bits.

**Returns**

*bytes* – Bit mask of states of the 4 digital output pins.

**get\_encoder\_counter**()

Get the encoder counter.

For devices that have an encoder, the current encoder position can be read.

**Returns**

*int* – The encoder count in encoder units.

**get\_front\_panel\_locked**()

Query if the device front panel locked.

**Returns**

*bool* – *True* if the device front panel is locked, *False* if not.

**get\_hardware\_info**()

Gets the hardware information from the device.

**Returns**

*structs.TLI\_HardwareInformation* – The hardware information.

**Raises**

*ThorlabsError* – If not successful.

**get\_hardware\_info\_block**()

Gets the hardware information in a block.

**Returns**

*structs.TLI\_HardwareInformation* – The hardware information.

**Raises**

*ThorlabsError* – If not successful.

**get\_homing\_params\_block**()

Get the homing parameters.



**Returns**

*structs.MOT\_HomingParameters* – The homing parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_homing\_velocity()**

Gets the homing velocity.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Returns**

*int* – The homing velocity in DeviceUnits (see manual).

**get\_hub\_bay()**

Gets the hub bay number this device is fitted to.

**Returns**

*bytes* – The number, 0x00 if unknown or 0xff if not on a hub.

**get\_jog\_mode()**

Gets the jog mode.

**Returns**

- *enums.MOT\_JogModes* – The jog mode.
- *enums.MOT\_StopModes* – The stop mode.

**Raises**

*ThorlabsError* – If not successful.

**get\_jog\_params\_block()**

Get the jog parameters.

**Returns**

*structs.MOT\_JogParameters* – The jog parameters.

**Raises**

*ThorlabsError* – 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 DeviceUnit to a RealValue.

**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 DeviceUnit to a RealValue.

**Returns**

- *int* – The maximum velocity in DeviceUnits (see manual).

- `int` – The acceleration in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – If not successful.

**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\_limit\_switch\_params()**

Gets the limit switch parameters.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**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**

*ThorlabsError* – If not successful.

**get\_limit\_switch\_params\_block()**

Get the limit switch parameters.

**Returns**

*structs.MOT\_LimitSwitchParameters* – The limit switch parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_mmi\_params()**

Get the MMI Parameters for the KCube Display Interface.

Deprecated calls by *get\_mmi\_params\_ext()*

**get\_mmi\_params\_block()**

Gets the MMI parameters for the device.

**Returns**

*structs.KMOT\_MMIParams* – The MMI parameters for the device.

**get\_mmi\_params\_ext()**

Get the MMI Parameters for the KCube Display Interface.

See `get_real_value_from_device_unit()` for converting from a `DeviceUnit` to a `RealValue`.

#### Returns

- `enums.KMOT_WheelMode` – The device joystick mode.
- `int` – The joystick maximum velocity in `DeviceUnits`.
- `int` – The joystick acceleration in `DeviceUnits`.
- `enums.KMOT_WheelDirectionSense` – The joystick direction sense.
- `int` – The first preset position in `DeviceUnits`.
- `int` – The second preset position in `DeviceUnits`.
- `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

`ThorlabsError` – 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

`ThorlabsError` – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**get\_move\_absolute\_position()**

Gets the move absolute position.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Returns**

*int* – The move 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 DeviceUnit to a RealValue.

**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**

`ThorlabsError` – If not successful.

**get\_number\_positions()**

Get the 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 `DeviceUnit` to a `RealValue`.

**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 `DeviceUnit` to a `RealValue`.

**Returns**

`int` – The position counter in `DeviceUnits` (see manual).

**get\_real\_value\_from\_device\_unit(device\_value, unit\_type)**

Converts a device value to a real-world value.

Either `load_settings()`, `load_named_settings()` or `set_motor_params_ext()` must be called before calling this function, otherwise the returned value will always be 0.

**Parameters**

- **device\_value** (`int`) – The device value.
- **unit\_type** (`enums.UnitType`) – The unit of the device value.

**Returns**

`float` – The real-world value.

**Raises**

`ThorlabsError` – 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 Stepper Motor maximum stage position.

See [\*get\\_real\\_value\\_from\\_device\\_unit\(\)\*](#) for converting from a DeviceUnit to a RealValue.

**Returns**

*int* – The maximum position in DeviceUnits (see manual).

**get\_stage\_axis\_min\_pos()**

Gets the Stepper Motor minimum stage position.

See [\*get\\_real\\_value\\_from\\_device\\_unit\(\)\*](#) for converting from a DeviceUnit to a RealValue.

**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 the polling functions, [\*start\\_polling\(\)\*](#).

**Returns**

*int* – The status bits from the device.

**get\_trigger\_config\_params()**

Get the Trigger Configuration Parameters.

**Returns**

- [\*enums.KMOT\\_TriggerPortMode\*](#) – The trigger 1 mode.
- [\*enums.KMOT\\_TriggerPortPolarity\*](#) – The trigger 1 polarity.
- [\*enums.KMOT\\_TriggerPortMode\*](#) – The trigger 2 mode.
- [\*enums.KMOT\\_TriggerPortPolarity\*](#) – The trigger 2 polarity.

**Raises**

[\*ThorlabsError\*](#) – If not successful.

**get\_trigger\_config\_params\_block()**

Gets the trigger configuration parameters block.

**Returns**

[\*structs.KMOT\\_TriggerConfig\*](#) – Options for controlling the trigger configuration.

**Raises**

[\*ThorlabsError\*](#) – If not successful.

**get\_trigger\_params\_params()**

Get the Trigger Parameters parameters.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**Returns**

- `int` – The trigger start position, forward, in DeviceUnits (see manual).
- `int` – The trigger interval, forward, in DeviceUnits (see manual).
- `int` – Number of trigger pulses, forward.
- `int` – The trigger start position, reverse, in DeviceUnits (see manual).
- `int` – The trigger interval, reverse, in DeviceUnits (see manual).
- `int` – Number of trigger pulses, reverse.
- `int` – Width of the trigger pulse in milliseconds, range 10 (10us) to 650000 (650ms).
- `int` – Number of cycles to perform triggering.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_trigger\_params\_params\_block()**

Gets the trigger parameters block.

**Returns**

[`structs.KMOT\_TriggerParams`](#) – Options for controlling the trigger.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_vel\_params()**

Gets the move velocity parameters.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**Returns**

- **max\_velocity** (`int`) – The maximum velocity in DeviceUnits (see manual).
- **acceleration** (`int`) – The acceleration in DeviceUnits (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_vel\_params\_block()**

Get the move velocity parameters.

**Returns**

[`structs.MOT\_VelocityParameters`](#) – The velocity parameters.

**Raises**

[`ThorlabsError`](#) – 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()**

Home the device.

Homing the device will set the device to a known state and determine the home position.

**Raises**

[ThorlabsError](#) – If not successful.

**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**

[ThorlabsError](#) – If not successful.

**load\_named\_settings(settings\_name)**

Update device with named settings.

**Parameters**

**settings\_name** (`str`) – The name of the device to load the settings for. Examples for the value of *setting\_name* can be found in *ThorlabsDefaultSettings.xml*, which gets created when the Kinesis software is installed.

**Raises**

[ThorlabsError](#) – 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**

[ThorlabsError](#) – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – If not successful.

**move\_relative(*displacement*)**

Move the motor by a relative amount.

See *get\_device\_unit\_from\_real\_value()* for converting from a RealValue to a DeviceUnit.

**Parameters**

**displacement** (*int*) – Signed displacement in DeviceUnits (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**move\_relative\_distance()**

Moves the device by a relative distance defined by *set\_move\_relative\_distance()*.

**Raises**

***ThorlabsError*** – If not successful.

**move\_to\_position(*index*)**

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 RealValue to a DeviceUnit.

**Parameters**

**index** (*int*) – The position in DeviceUnits (see manual).

**Raises**

***ThorlabsError*** – 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**

***ThorlabsError*** – If not successful.

**persist\_settings()**

Persist the devices current settings.

**Raises**

*ThorlabsError* – 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** (*MotionControlCallback*) – A function to be called whenever messages are received.

**request\_backlash()**

Requests the backlash.

**Raises**

*ThorlabsError* – If not successful.

**request\_dcpid\_params()**

Request the PID parameters for DC motors used in an algorithm involving calculus.

**Raises**

*ThorlabsError* – If not successful.

**request\_digital\_outputs()**

Requests the digital output bits.

**Raises**

*ThorlabsError* – If not successful.

**request\_encoder\_counter()**

Requests the encoder counter.

**Raises**

*ThorlabsError* – If not successful.

**request\_front\_panel\_locked()**

Ask the device if its front panel is locked.

**Raises**

*ThorlabsError* – If not successful.

**request\_homing\_params()**

Requests the homing parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_jog\_params()**

Requests the jog parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_led\_switches()**

Requests the LED indicator bits on the cube.

**Raises**

*ThorlabsError* – If not successful.

**request\_limit\_switch\_params()**

Requests the limit switch parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_mmi\_params()**

Requests the MMI Parameters for the KCube Display Interface.

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_absolute\_position()**

Requests the position of next absolute move.

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_relative\_distance()**

Requests the relative move distance.

**Raises**

*ThorlabsError* – If not successful.

**request\_pos\_trigger\_params()**

Requests the position trigger parameters.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

[ThorlabsError](#) – If not successful.

**request\_trigger\_config\_params()**

Requests the Trigger Configuration Parameters.

**Raises**

[ThorlabsError](#) – If not successful.

**request\_vel\_params()**

Requests the velocity parameters.

**Raises**

[ThorlabsError](#) – If not successful.

**reset\_rotation\_modes()**

Reset the rotation modes for a rotational device.

**Raises**

[ThorlabsError](#) – If not successful.

**reset\_stage\_to\_defaults()**

Reset the stage settings to defaults.

**Raises**

[ThorlabsError](#) – If not successful.

**resume\_move\_messages()**

Resume suspended move messages.

**Raises**

[ThorlabsError](#) – 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 RealValue to a DeviceUnit.

**Parameters**

**distance** ([int](#)) – The backlash distance in DeviceUnits (see manual).

**Raises**

[ThorlabsError](#) – If not successful.

**set\_dcpid\_params(*params*)**

Set the PID parameters for DC motors used in an algorithm involving calculus.

**Parameters**

**params** ([structs.MOT\\_DC\\_PIDParameters](#)) – The DC PID parameters.

**Raises**

- [ThorlabsError](#) – If not successful.

- **TypeError** – If the data type of *params* is not *structs.MOT\_DC\_PIDParameters*

**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**

**ThorlabsError** – If not successful.

**set\_direction**(*reverse*)

Sets the motor direction sense.

This function is used because some actuators 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.

**Raises**

**ThorlabsError** – If not successful.

**set\_encoder\_counter**(*count*)

Set the Encoder Counter values.

Setting the encoder counter to zero, effectively defines a home position on the encoder strip. Note, setting this value does not move the device.

**Parameters**

**count** (*int*) – The encoder count in encoder units.

**Raises**

**ThorlabsError** – If not successful.

**set\_front\_panel\_lock**(*locked*)

Sets the device front panel lock state.

**Parameters**

**locked** (*bool*) – *True* to lock the device, *False* to unlock

**Raises**

**ThorlabsError** – If not successful.

**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**

*ThorlabsError* – If not successful.

**set\_homing\_velocity**(*velocity*)

Sets the homing velocity.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

**velocity** (*int*) – The homing velocity in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**set\_jog\_params\_block**(*jog\_params*)

Set the jog parameters.

**Parameters**

**jog\_params** (*structs.MOT\_JogParameters*) – The jog parameters.

**Raises**

- *ThorlabsError* – 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 *RealValue* to a *DeviceUnit*.

**Parameters**

**step\_size** (*int*) – The step size in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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 *RealValue* to a *DeviceUnit*.

**Parameters**

- **max\_velocity** (*int*) – The maximum velocity in DeviceUnits (see manual).
- **acceleration** (*int*) – The acceleration in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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 RealValue to a DeviceUnit.

**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**

*ThorlabsError* – If not successful.

**set\_limit\_switch\_params\_block**(*params*)

Set the limit switch parameters.

**Parameters**

**params** (*structs.MOT\_LimitSwitchParameters*) – The limit switch parameters.

**Raises**

*ThorlabsError* – 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\_mmi\_params**(*joystick\_mode, joystick\_max\_velocity, joystick\_acceleration, direction\_sense, preset\_position1, preset\_position2, display\_intensity*)

Set the MMI Parameters for the KCube Display Interface.

Deprecated calls *set\_mmi\_params\_ext()* setting the *display\_timeout* to 1 minute and the *display\_dim\_intensity* to 8.

**Raises**

*ThorlabsError* – If not successful.

**set\_mmi\_params\_block**(*mmi\_params*)

Sets the MMI parameters for the device.

**Parameters**

**mmi\_params** (*structs.KMOT\_MMIParams*) – Options for controlling the mmi.

**Raises**

*ThorlabsError* – If not successful.

**set\_mmi\_params\_ext**(*joystick\_mode, joystick\_max\_velocity, joystick\_acceleration, direction\_sense, preset\_position1, preset\_position2, display\_intensity, display\_timeout, display\_dim\_intensity*)

Set the MMI Parameters for the KCube Display Interface.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

- **joystick\_mode** (*enums.KMOT\_WheelMode*) – The device joystick mode as a *enums.KMOT\_WheelMode* enum value or member name.
- **joystick\_max\_velocity** (*int*) – The joystick maximum velocity in DeviceUnits.
- **joystick\_acceleration** (*int*) – The joystick acceleration in DeviceUnits.
- **direction\_sense** (*enums.KMOT\_WheelDirectionSense*) – The joystick direction sense as a *enums.KMOT\_WheelDirectionSense* enum value or member name.
- **preset\_position1** (*int*) – The first preset position in DeviceUnits.
- **preset\_position2** (*int*) – The second preset position in DeviceUnits.
- **display\_intensity** (*int*) – The display intensity, range 0 to 100%.



- **display\_timeout** (*int*) – The display timeout, range 0 to 480 in minutes (0 is off, otherwise the inactivity period before dimming the display).
- **display\_dim\_intensity** (*int*) – The display dimmed intensity, range 0 to 10 (after the timeout period the device display will dim).

**Raises**

*ThorlabsError* – If not successful.

**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 *DeviceUnit* to a *RealValue*.

**Parameters**

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises**

*ThorlabsError* – 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 *DeviceUnit* to a *RealValue*.

**Parameters**

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises**

*ThorlabsError* – 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 *DeviceUnit* to a *RealValue*.

**Parameters**

- **min\_position** (`float`) – The minimum position in `RealWorldUnits` [millimeters or degrees].
- **max\_position** (`float`) – The maximum position in `RealWorldUnits` [millimeters or degrees].

**Raises**

[`ThorlabsError`](#) – 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**

[`ThorlabsError`](#) – 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 `DeviceUnit` to a `RealValue`.

**Parameters**

- **max\_velocity** (`float`) – The maximum velocity in `RealWorldUnits` [millimeters or degrees].
- **max\_acceleration** (`float`) – The maximum acceleration in `RealWorldUnits` [millimeters or degrees].

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_move\_absolute\_position**(*position*)

Sets the move absolute position.

See [`get\_device\_unit\_from\_real\_value\(\)`](#) for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

**position** (`int`) – The absolute position in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_move\_relative\_distance**(*distance*)

Sets the move relative distance.

See [`get\_device\_unit\_from\_real\_value\(\)`](#) for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

**distance** (`int`) – The relative position in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – 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 `RealValue` to a `DeviceUnit`.

**Parameters**

**count** (`int`) – The position counter in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_rotation\_modes**(*mode*, *direction*)

Set the rotation modes for a rotational device.

**Parameters**

- **mode** ([`enums.MOT\_MovementModes`](#)) – The travel mode as a [`enums.MOT\_MovementModes`](#) enum value or member name.
- **direction** ([`enums.MOT\_MovementDirections`](#)) – The travel mode as a [`enums.MOT\_MovementDirections`](#) enum value or member name.

**Raises**

[`ThorlabsError`](#) – 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 `RealValue` to a `DeviceUnit`.

**Parameters**

- **min\_position** (`int`) – The minimum position in `DeviceUnits` (see manual).
- **max\_position** (`int`) – The maximum position in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – If not successful.

**set\_trigger\_config\_params**(*mode1*, *polarity1*, *mode2*, *polarity2*)

Set the trigger configuration parameters.

**Parameters**

- **mode1** ([`enums.KMOT\_TriggerPortMode`](#)) – The trigger 1 mode as a [`KMOT\_TriggerPortMode`](#) enum value or member name.
- **polarity1** ([`enums.KMOT\_TriggerPortPolarity`](#)) – The trigger 1 polarity as a [`KMOT\_TriggerPortPolarity`](#) enum value or member name.
- **mode2** ([`enums.KMOT\_TriggerPortMode`](#)) – The trigger 2 mode as a [`KMOT\_TriggerPortMode`](#) enum value or member name.

- **polarity2** (*enums.KMOT\_TriggerPortPolarity*) – The trigger 2 polarity as a *KMOT\_TriggerPortPolarity* enum value or member name.

**Raises**

*ThorlabsError* – If not successful.

**set\_trigger\_config\_params\_block**(*trigger\_config\_params*)

Sets the trigger configuration parameters block.

**Parameters**

**trigger\_config\_params** (*structs.KMOT\_TriggerConfig*) – Options for controlling the trigger configuration.

**Raises**

- *ThorlabsError* – If not successful.
- *TypeError* – If *trigger\_config\_params* is not a *structs.KMOT\_TriggerConfig*.

**set\_trigger\_params\_params**(*trigger\_start\_position\_fwd*, *trigger\_interval\_fwd*,  
*trigger\_pulse\_count\_fwd*, *trigger\_start\_position\_rev*,  
*trigger\_interval\_rev*, *trigger\_pulse\_count\_rev*,  
*trigger\_pulse\_width*, *cycle\_count*)

Set the Trigger Parameters parameters.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Parameters**

- **trigger\_start\_position\_fwd** (*int*) – The trigger start position, forward, in DeviceUnits (see manual).
- **trigger\_interval\_fwd** (*int*) – The trigger interval, forward, in DeviceUnits (see manual).
- **trigger\_pulse\_count\_fwd** (*int*) – Number of trigger pulses, forward.
- **trigger\_start\_position\_rev** (*int*) – The trigger start position, reverse, in DeviceUnits (see manual).
- **trigger\_interval\_rev** (*int*) – The trigger interval, reverse, in DeviceUnits (see manual).
- **trigger\_pulse\_count\_rev** (*int*) – Number of trigger pulses., reverse.
- **trigger\_pulse\_width** (*int*) – Width of the trigger pulse in milliseconds, range 10 (10us) to 650000 (650ms).
- **cycle\_count** (*int*) – Number of cycles to perform triggering.

**Raises**

*ThorlabsError* – If not successful.

**set\_trigger\_params\_params\_block**(*trigger\_params\_params*)

Set the Trigger Parameters parameters.

**Parameters**

**trigger\_params\_params** (*structs.KMOT\_TriggerParams*) – Options for controlling the trigger.

**Raises**

- **ThorlabsError** – If not successful.
- **TypeError** – If *trigger\_params\_params* is not a *structs.KMOT\_TriggerParams*.

**set\_vel\_params**(*max\_velocity*, *acceleration*)

Sets the move velocity parameters.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

- **max\_velocity** (*int*) – The maximum velocity in *DeviceUnits* (see manual).
- **acceleration** (*int*) – The acceleration in *DeviceUnits* (see manual).

**Raises**

**ThorlabsError** – 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**

**ThorlabsError** – 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**

**ThorlabsError** – If not successful.

**stop\_immediate**()

Stop the current move immediately (with the risk of losing track of the position).

**Raises**

**ThorlabsError** – If not successful.

**stop\_polling**()

Stops the internal polling loop.

**stop\_profiled()**

Stop the current move using the current velocity profile.

**Raises**

*ThorlabsError* – If not successful.

**suspend\_move\_messages()**

Suspend automatic messages at ends of moves.

Useful to speed up part of real-time system with lots of short moves.

**Raises**

*ThorlabsError* – 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.
- *bool* – *True* if monitoring is enabled otherwise *False*.

**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**

*ThorlabsError* – 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: *MotionControl*

A wrapper around `Thorlabs.MotionControl.KCube.Solenoid.dll`.

The *properties* for a KCubeSolenoid connection supports the following key-value pairs in the *Connections Database*:

```
'device_name': str, the device name found in ThorlabsDefaultSettings.xml.  
↪ [default: None]
```

Do not instantiate this class directly. Use the *connect()* method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A 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**

**ThorlabsError** – If not successful.

**get\_cycle\_params\_block()**

Get the cycle parameters.

**Returns**

**structs.SC\_CycleParameters** – The cycle parameters.

**Raises**

**ThorlabsError** – 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**

*ThorlabsError* – If not successful.

**get\_hardware\_info\_block()**

Gets the hardware information in a block.

**Returns**

*structs.TLI\_HardwareInformation* – The hardware information.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**get\_mmi\_params\_block()**

Gets the MMI parameters for the device.

**Warning:** This function is currently not in the DLL, as of v1.11.2, but it is in the header file.

**Returns**

*structs.KSC\_MMIParams* – Options for controlling the MMI.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

`ThorlabsError` – 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**

`ThorlabsError` – 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**

`ThorlabsError` – If not successful.

**load\_named\_settings(settings\_name)**

Update device with named settings.

**Parameters**

**settings\_name** (`str`) – The name of the device to load the settings for. Examples for the value of `setting_name` can be found in `ThorlabsDefaultSettings.xml`, which gets created when the Kinesis software is installed.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**persist\_settings()**

Persist the devices current settings.

**Raises**

*ThorlabsError* – 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** (*MotionControlCallback*) – A function to be called whenever messages are received.

**request\_cycle\_params()**

Requests the cycle parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_digital\_outputs()**

Requests the digital output bits.

**Raises**

*ThorlabsError* – If not successful.

**request\_hub\_bay()**

Requests the hub bay number this device is fitted to.

**Raises**

*ThorlabsError* – If not successful.

**request\_led\_switches()**

Requests the LED indicator bits on the cube.

**Raises**

*ThorlabsError* – If not successful.

**request\_mmi\_params()**

Requests the MMI Parameters for the KCube Display Interface.

**Raises**

*ThorlabsError* – If not successful.

**request\_operating\_mode()**

Requests the operating mode.

**Raises**

*ThorlabsError* – If not successful.

**request\_operating\_state()**

Requests the operating state.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**request\_trigger\_config\_params()**

Requests the Trigger Configuration Parameters.

**Raises**

*ThorlabsError* – 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**

***ThorlabsError*** – If not successful.

**set\_cycle\_params\_block**(*cycle\_params*)

Sets the cycle parameters.

**Parameters**

**cycle\_params** (*structs.SC\_CycleParameters*) – The new cycle parameters.

**Raises**

***ThorlabsError*** – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – If not successful.

**set\_mmi\_params\_block**(*mmi\_params*)

Sets the MMI parameters for the device.

**Warning:** This function is currently not in the DLL, as of v1.11.2, but it is in the header file.

**Parameters**

**mmi\_params** (*structs.KSC\_MMIParams*) – Options for controlling the MMI.

**Raises**

***ThorlabsError*** – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – 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**

***ThorlabsError*** – If not successful.

**set\_trigger\_config\_params**(*mode1*, *polarity1*, *mode2*, *polarity2*)

Set the trigger configuration parameters.

**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**

***ThorlabsError*** – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

## **msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor module**

This module provides all the functionality required to control a KCube Stepper Motor (KST101).

**class** msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.**KCubeStepperMotor**(*record*)

Bases: *MotionControl*

A wrapper around `Thorlabs.MotionControl.KCube.StepperMotor.dll`.

The *properties* for a KCubeStepperMotor connection supports the following key-value pairs in the *Connections Database*:

```
'device_name': str, the device name found in ThorlabsDefaultSettings.xml.  
↪ [default: None]
```

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

**Parameters**

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

**can\_device\_lock\_front\_panel()**

Determine if the device front panel can be locked.

**Returns**

*bool* – *True* if the front panel of the device can be locked, *False* if not.

**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**

*ThorlabsError* – 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**

*ThorlabsError* – 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 `DeviceUnit` to a `RealValue`.

**Returns**

`int` – The backlash distance in `DeviceUnits` (see manual).

**get\_bow\_index()**

Gets the stepper motor bow index.

**Returns**

`int` – The bow index.

**get\_calibration\_file()**

Get calibration file for this motor.

**Returns**

`str` – The filename of the calibration file.

**Raises**

`ThorlabsError` – 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()`, `load_named_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** (`enums.UnitType`) – The unit of the real-world value.

**Returns**

`int` – The device value.

**Raises**

`ThorlabsError` – If not successful.

**get\_digital\_outputs()**

Gets the digital output bits.

**Returns**

`bytes` – Bit mask of states of the 4 digital output pins.

**get\_encoder\_counter()**

Get the encoder counter.

For devices that have an encoder, the current encoder position can be read.

**Returns**

`int` – The encoder count in encoder units.

**get\_front\_panel\_locked()**

Query if the device front panel locked.

**Returns**

`bool` – `True` if the device front panel is locked, `False` if not.

**get\_hardware\_info()**

Gets the hardware information from the device.

**Returns**

`structs.TLI_HardwareInformation` – The hardware information.

**Raises**

`ThorlabsError` – If not successful.

**get\_hardware\_info\_block()**

Gets the hardware information in a block.

**Returns**

`structs.TLI_HardwareInformation` – The hardware information.

**Raises**

`ThorlabsError` – If not successful.

**get\_homing\_params\_block()**

Get the homing parameters.

**Returns**

`structs.MOT_HomingParameters` – The homing parameters.

**Raises**

`ThorlabsError` – If not successful.

**get\_homing\_velocity()**

Gets the homing velocity.

See `get_real_value_from_device_unit()` for converting from a `DeviceUnit` to a `RealValue`.

**Returns**

`int` – The homing velocity in `DeviceUnits` (see manual).

**get\_hub\_bay()**

Gets the hub bay number this device is fitted to.

**Returns**

`bytes` – The number, `0x00` if unknown or `0xff` if not on a hub.

**get\_jog\_mode()**

Gets the jog mode.

**Returns**

- `enums.MOT_JogModes` – The jog mode.
- `enums.MOT_StopModes` – The stop mode.

**Raises**

*ThorlabsError* – If not successful.

**get\_jog\_params\_block()**

Get the jog parameters.

**Returns**

*structs.MOT\_JogParameters* – The jog parameters.

**Raises**

*ThorlabsError* – 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 DeviceUnit to a RealValue.

**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 DeviceUnit to a RealValue.

**Returns**

- *int* – The maximum velocity in DeviceUnits (see manual).
- *int* – The acceleration in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – If not successful.

**get\_limit\_switch\_params()**

Gets the limit switch parameters.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**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**

*ThorlabsError* – If not successful.

**get\_limit\_switch\_params\_block()**

Get the limit switch parameters.

**Returns**

*structs.MOT\_LimitSwitchParameters* – The limit switch parameters.

**Raises**

*ThorlabsError* – If not successful.

**get\_mmi\_params()**

Get the MMI Parameters for the KCube Display Interface.

Deprecated calls by *get\_mmi\_params\_ext()*

**get\_mmi\_params\_block()**

Gets the MMI parameters for the device.

**Returns**

*structs.KMOT\_MMIParams* – The MMI parameters for the device.

**get\_mmi\_params\_ext()**

Get the MMI Parameters for the KCube Display Interface.

See *get\_real\_value\_from\_device\_unit()* for converting from a DeviceUnit to a RealValue.

**Returns**

- *enums.KMOT\_WheelMode* – The device joystick mode.
- *int* – The joystick maximum velocity in DeviceUnits.
- *int* – The joystick acceleration in DeviceUnits.
- *enums.KMOT\_WheelDirectionSense* – The joystick direction sense.
- *int* – The first preset position in DeviceUnits.
- *int* – The second preset position in DeviceUnits.
- *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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**get\_move\_absolute\_position()**

Gets the move absolute position.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a `DeviceUnit` to a `RealValue`.

**Returns**

`int` – The move 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 `DeviceUnit` to a `RealValue`.

**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**

[`ThorlabsError`](#) – If not successful.

**get\_number\_positions()**

Get the 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\_pid\_loop\_encoder\_coeff()**

Gets the encoder PID loop coefficient.

This is the encoder coefficient. Use 0.0 to disable the encoder or if no encoder is present otherwise a positive encoder coefficient.

**Returns**

`float` – The encoder PID loop coefficient.

**get\_pid\_loop\_encoder\_params()**

Gets the Encoder PID loop parameters.

**Returns**

[`structs.MOT\_PIDLoopEncoderParams`](#) – The Encoder PID loop parameters.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_position()**

Get the current position.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**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 DeviceUnit to a RealValue.

**Returns**

[`int`](#) – The position counter in DeviceUnits (see manual).

**get\_power\_params()**

Gets the power parameters for the stepper motor.

**Returns**

[`structs.MOT\_PowerParameters`](#) – The power parameters.

**Raises**

[`ThorlabsError`](#) – 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\(\)`](#), [`load\_named\_settings\(\)`](#) or [`set\_motor\_params\_ext\(\)`](#) must be called before calling this function, otherwise the returned value will always be 0.

**Parameters**

- **device\_value** ([`int`](#)) – The device value.
- **unit\_type** ([`enums.UnitType`](#)) – The unit of the device value.

**Returns**

[`float`](#) – The real-world value.

**Raises**

[`ThorlabsError`](#) – 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 Stepper Motor maximum stage position.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**Returns**

`int` – The maximum position in DeviceUnits (see manual).

**get\_stage\_axis\_min\_pos()**

Gets the Stepper Motor minimum stage position.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**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 the polling functions, [`start\_polling\(\)`](#).

**Returns**

`int` – The status bits from the device.

**get\_trigger\_config\_params()**

Get the Trigger Configuration Parameters.

**Returns**

- [`enums.KMOT\_TriggerPortMode`](#) – The trigger 1 mode.
- [`enums.KMOT\_TriggerPortPolarity`](#) – The trigger 1 polarity.
- [`enums.KMOT\_TriggerPortMode`](#) – The trigger 2 mode.
- [`enums.KMOT\_TriggerPortPolarity`](#) – The trigger 2 polarity.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_trigger\_config\_params\_block()**

Gets the trigger configuration parameters block.

**Returns**

[`structs.KMOT\_TriggerConfig`](#) – Options for controlling the trigger configuration.

**Raises**

[`ThorlabsError`](#) – If not successful.

**get\_trigger\_params\_params()**

Get the Trigger Parameters parameters.

See [`get\_real\_value\_from\_device\_unit\(\)`](#) for converting from a DeviceUnit to a RealValue.

**Returns**



- `int` – The trigger start position, forward, in `DeviceUnits` (see manual).
- `int` – The trigger interval, forward, in `DeviceUnits` (see manual).
- `int` – Number of trigger pulses, forward.
- `int` – The trigger start position, reverse, in `DeviceUnits` (see manual).
- `int` – The trigger interval, reverse, in `DeviceUnits` (see manual).
- `int` – Number of trigger pulses., reverse.
- `int` – Width of the trigger pulse in milliseconds, range 10 (10us) to 650000 (650ms).
- `int` – Number of cycles to perform triggering.

**Raises**

***ThorlabsError*** – If not successful.

**get\_trigger\_params\_params\_block()**

Gets the trigger parameters block.

**Returns**

***structs.KMOT\_TriggerParams*** – Options for controlling the trigger.

**Raises**

***ThorlabsError*** – If not successful.

**get\_vel\_params()**

Gets the move velocity parameters.

See ***get\_real\_value\_from\_device\_unit()*** for converting from a `DeviceUnit` to a `RealValue`.

**Returns**

- **max\_velocity** (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**get\_vel\_params\_block()**

Get the move velocity parameters.

**Returns**

***structs.MOT\_VelocityParameters*** – The velocity parameters.

**Raises**

***ThorlabsError*** – 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()**

Home the device.

Homing the device will set the device to a known state and determine the home position.

**Raises**

**ThorlabsError** – 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**

**ThorlabsError** – If not successful.

**load\_named\_settings(settings\_name)**

Update device with named settings.

**Parameters**

**settings\_name** (**str**) – The name of the device to load the settings for. Examples for the value of *setting\_name* can be found in *ThorlabsDefaultSettings.xml*, which gets created when the Kinesis software is installed.

**Raises**

**ThorlabsError** – 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**

**ThorlabsError** – 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**

[\*ThorlabsError\*](#) – 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**

[\*ThorlabsError\*](#) – If not successful.

**move\_relative**(*displacement*)

Move the motor by a relative amount.

See [\*get\\_device\\_unit\\_from\\_real\\_value\(\)\*](#) for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

**displacement** ([\*int\*](#)) – Signed displacement in *DeviceUnits* (see manual).

**Raises**

[\*ThorlabsError\*](#) – If not successful.

**move\_relative\_distance**()

Moves the device by a relative distance defined by [\*set\\_move\\_relative\\_distance\(\)\*](#).

**Raises**

[\*ThorlabsError\*](#) – If not successful.

**move\_to\_position**(*index*)

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 *RealValue* to a *DeviceUnit*.

**Parameters**

**index** ([\*int\*](#)) – The position in *DeviceUnits* (see manual).

**Raises**

[\*ThorlabsError\*](#) – 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**

[\*ThorlabsError\*](#) – If not successful.

**persist\_settings()**

Persist the devices current settings.

**Raises**

*ThorlabsError* – 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** (*MotionControlCallback*) – A function to be called whenever messages are received.

**request\_backlash()**

Requests the backlash.

**Raises**

*ThorlabsError* – If not successful.

**request\_bow\_index()**

Requests the stepper motor bow index.

**Raises**

*ThorlabsError* – If not successful.

**request\_digital\_outputs()**

Requests the digital output bits.

**Raises**

*ThorlabsError* – If not successful.

**request\_encoder\_counter()**

Requests the encoder counter.

**Raises**

*ThorlabsError* – If not successful.

**request\_front\_panel\_locked()**

Ask the device if its front panel is locked.

**Raises**

*ThorlabsError* – If not successful.

**request\_homing\_params()**

Requests the homing parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_jog\_params()**

Requests the jog parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_limit\_switch\_params()**

Requests the limit switch parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_mmi\_params()**

Requests the MMI Parameters for the KCube Display Interface.

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_absolute\_position()**

Requests the position of next absolute move.

**Raises**

*ThorlabsError* – If not successful.

**request\_move\_relative\_distance()**

Requests the relative move distance.

**Raises**

*ThorlabsError* – If not successful.

**request\_pid\_loop\_encoder\_params()**

Requests the Encoder PID loop parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_pos\_trigger\_params()**

Requests the position trigger parameters.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**request\_power\_params()**

Requests the power parameters.

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**request\_trigger\_config\_params()**

Requests the Trigger Configuration Parameters.

**Raises**

*ThorlabsError* – If not successful.

**request\_vel\_params()**

Requests the velocity parameters.

**Raises**

*ThorlabsError* – If not successful.

**reset\_rotation\_modes()**

Reset the rotation modes for a rotational device.

**Raises**

*ThorlabsError* – If not successful.

**resume\_move\_messages()**

Resume suspended move messages.

**Raises**

*ThorlabsError* – 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 RealValue to a DeviceUnit.

**Parameters**

**distance** (*int*) – The backlash distance in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_bow\_index(*bow\_index*)**

Sets the stepper motor bow index.

**Parameters**

**bow\_index** (*int*) – The bow index.

**Raises**

*ThorlabsError* – 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**

*OSError* – If the *path* does not exist.

**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**

*ThorlabsError* – If not successful.

**set\_direction**(*reverse*)

Sets the motor direction sense.

This function is used because some actuators 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.

**Raises**

*ThorlabsError* – If not successful.

**set\_encoder\_counter**(*count*)

Set the Encoder Counter values.

Setting the encoder counter to zero, effectively defines a home position on the encoder strip. Note, setting this value does not move the device.

**Parameters**

**count** (*int*) – The encoder count in encoder units.

**Raises**

*ThorlabsError* – If not successful.

**set\_front\_panel\_lock**(*locked*)

Sets the device front panel lock state.

**Parameters**

**locked** (*bool*) – *True* to lock the device, *False* to unlock

**Raises**

*ThorlabsError* – If not successful.

**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**

*ThorlabsError* – If not successful.

**set\_homing\_velocity**(*velocity*)

Sets the homing velocity.

See *get\_device\_unit\_from\_real\_value()* for converting from a *RealValue* to a *DeviceUnit*.

**Parameters**

**velocity** (*int*) – The homing velocity in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**set\_jog\_params\_block**(*jog\_params*)

Set the jog parameters.

**Parameters**

**jog\_params** (*structs.MOT\_JogParameters*) – The jog parameters.

**Raises**

- *ThorlabsError* – 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 *RealValue* to a *DeviceUnit*.

**Parameters**

**step\_size** (*int*) – The step size in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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 `RealValue` to a `DeviceUnit`.

**Parameters**

- **max\_velocity** (`int`) – The maximum velocity in `DeviceUnits` (see manual).
- **acceleration** (`int`) – The acceleration in `DeviceUnits` (see manual).

**Raises**

[`ThorlabsError`](#) – 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 `RealValue` to a `DeviceUnit`.

**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**

[`ThorlabsError`](#) – If not successful.

**set\_limit\_switch\_params\_block**(*params*)

Set the limit switch parameters.

**Parameters**

**params** (`structs.MOT_LimitSwitchParameters`) – The limit switch parameters.

**Raises**

[`ThorlabsError`](#) – 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\_mmi\_params**(*joystick\_mode, joystick\_max\_velocity, joystick\_acceleration, direction\_sense, preset\_position1, preset\_position2, display\_intensity*)

Set the MMI Parameters for the KCube Display Interface.

Deprecated calls **set\_mmi\_params\_ext()** setting the *display\_timeout* to 1 minute and the *display\_dim\_intensity* to 8.

**Raises**

**ThorlabsError** – If not successful.

**set\_mmi\_params\_block**(*mmi\_params*)

Sets the MMI parameters for the device.

**Parameters**

**mmi\_params** (*structs.KMOT\_MMIParams*) – Options for controlling the mmi.

**Raises**

**ThorlabsError** – If not successful.

**set\_mmi\_params\_ext**(*joystick\_mode, joystick\_max\_velocity, joystick\_acceleration, direction\_sense, preset\_position1, preset\_position2, display\_intensity, display\_timeout, display\_dim\_intensity*)

Set the MMI Parameters for the KCube Display Interface.

See **get\_real\_value\_from\_device\_unit()** for converting from a DeviceUnit to a RealValue.

**Parameters**

- **joystick\_mode** (*enums.KMOT\_WheelMode*) – The device joystick mode as a *enums.KMOT\_WheelMode* enum value or member name.
- **joystick\_max\_velocity** (*int*) – The joystick maximum velocity in DeviceUnits.
- **joystick\_acceleration** (*int*) – The joystick acceleration in DeviceUnits.
- **direction\_sense** (*enums.KMOT\_WheelDirectionSense*) – The joystick direction sense as a *enums.KMOT\_WheelDirectionSense* enum value or member name.
- **preset\_position1** (*int*) – The first preset position in DeviceUnits.
- **preset\_position2** (*int*) – The second preset position in DeviceUnits.
- **display\_intensity** (*int*) – The display intensity, range 0 to 100%.
- **display\_timeout** (*int*) – The display timeout, range 0 to 480 in minutes (0 is off, otherwise the inactivity period before dimming the display).

- **display\_dim\_intensity** (*int*) – The display dimmed intensity, range 0 to 10 (after the timeout period the device display will dim).

**Raises**

*ThorlabsError* – If not successful.

**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 *DeviceUnit* to a *RealValue*.

**Parameters**

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises**

*ThorlabsError* – 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 *DeviceUnit* to a *RealValue*.

**Parameters**

- **steps\_per\_rev** (*float*) – The steps per revolution.
- **gear\_box\_ratio** (*float*) – The gear box ratio.
- **pitch** (*float*) – The pitch.

**Raises**

*ThorlabsError* – 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 *DeviceUnit* to a *RealValue*.

**Parameters**

- **min\_position** (*float*) – The minimum position in *RealWorldUnits* [millimeters or degrees].

- **max\_position** (*float*) – The maximum position in `RealWorldUnits` [millimeters or degrees].

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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 `DeviceUnit` to a `RealValue`.

**Parameters**

- **max\_velocity** (*float*) – The maximum velocity in `RealWorldUnits` [millimeters or degrees].
- **max\_acceleration** (*float*) – The maximum acceleration in `RealWorldUnits` [millimeters or degrees].

**Raises**

*ThorlabsError* – If not successful.

**set\_move\_absolute\_position**(*position*)

Sets the move absolute position.

See *get\_device\_unit\_from\_real\_value()* for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

**position** (*int*) – The absolute position in `DeviceUnits` (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_move\_relative\_distance**(*distance*)

Sets the move relative distance.

See *get\_device\_unit\_from\_real\_value()* for converting from a `RealValue` to a `DeviceUnit`.

**Parameters**

**distance** (*int*) – The relative position in `DeviceUnits` (see manual).

**Raises**

*ThorlabsError* – If not successful.

**set\_pid\_loop\_encoder\_coeff**(*coeff*)

Sets the encoder PID loop coefficient.

This is the encoder coefficient. Use 0.0 to disable the encoder or if no encoder is present otherwise a positive encoder coefficient.

**Parameters**

**coeff** (*float*) – The encoder PID loop coefficient.

**Raises**

*ThorlabsError* – If not successful.

**set\_pid\_loop\_encoder\_params**(*params*)

Sets the encoder PID loop parameters.

**Parameters**

**params** (*structs.MOT\_PIDLoopEncoderParams*) – The encoder PID loop parameters.

**Raises**

- *ThorlabsError* – If not successful.
- *TypeError* – If *params* is not a *structs.MOT\_PIDLoopEncoderParams*.

**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 *RealValue* to a *DeviceUnit*.

**Parameters**

**count** (*int*) – The position counter in *DeviceUnits* (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**set\_rotation\_modes**(*mode, direction*)

Set the rotation modes for a rotational device.

**Parameters**

- **mode** (*enums.MOT\_MovementModes*) – The travel mode as a *enums.MOT\_MovementModes* enum value or member name.
- **direction** (*enums.MOT\_MovementDirections*) – The travel mode as a *enums.MOT\_MovementDirections* enum value or member name.

**Raises**

***ThorlabsError*** – 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 *RealValue* to a *DeviceUnit*.

**Parameters**

- **min\_position** (*int*) – The minimum position in *DeviceUnits* (see manual).
- **max\_position** (*int*) – The maximum position in *DeviceUnits* (see manual).

**Raises**

***ThorlabsError*** – If not successful.

**set\_trigger\_config\_params**(*mode1*, *polarity1*, *mode2*, *polarity2*)

Set the trigger configuration parameters.

**Parameters**

- **mode1** (*enums.KMOT\_TriggerPortMode*) – The trigger 1 mode as a *KMOT\_TriggerPortMode* enum value or member name.
- **polarity1** (*enums.KMOT\_TriggerPortPolarity*) – The trigger 1 polarity as a *KMOT\_TriggerPortPolarity* enum value or member name.
- **mode2** (*enums.KMOT\_TriggerPortMode*) – The trigger 2 mode as a *KMOT\_TriggerPortMode* enum value or member name.
- **polarity2** (*enums.KMOT\_TriggerPortPolarity*) – The trigger 2 polarity as a *KMOT\_TriggerPortPolarity* enum value or member name.

**Raises**

***ThorlabsError*** – If not successful.

**set\_trigger\_config\_params\_block**(*trigger\_config\_params*)

Sets the trigger configuration parameters block.

**Parameters**

**trigger\_config\_params** (*structs.KMOT\_TriggerConfig*) – Options for controlling the trigger configuration.

**Raises**

- ***ThorlabsError*** – If not successful.
- ***TypeError*** – If *trigger\_config\_params* is not a *structs.KMOT\_TriggerConfig*.

**set\_trigger\_params\_params**(*trigger\_start\_position\_fwd*, *trigger\_interval\_fwd*,  
*trigger\_pulse\_count\_fwd*, *trigger\_start\_position\_rev*,  
*trigger\_interval\_rev*, *trigger\_pulse\_count\_rev*,  
*trigger\_pulse\_width*, *cycle\_count*)

Set the Trigger Parameters parameters.

See `get_real_value_from_device_unit()` for converting from a DeviceUnit to a RealValue.

#### Parameters

- **trigger\_start\_position\_fwd** (`int`) – The trigger start position, forward, in DeviceUnits (see manual).
- **trigger\_interval\_fwd** (`int`) – The trigger interval, forward, in DeviceUnits (see manual).
- **trigger\_pulse\_count\_fwd** (`int`) – Number of trigger pulses, forward.
- **trigger\_start\_position\_rev** (`int`) – The trigger start position, reverse, in DeviceUnits (see manual).
- **trigger\_interval\_rev** (`int`) – The trigger interval, reverse, in DeviceUnits (see manual).
- **trigger\_pulse\_count\_rev** (`int`) – Number of trigger pulses, reverse.
- **trigger\_pulse\_width** (`int`) – Width of the trigger pulse in milliseconds, range 10 (10us) to 650000 (650ms).
- **cycle\_count** (`int`) – Number of cycles to perform triggering.

#### Raises

**ThorlabsError** – If not successful.

**set\_trigger\_params\_params\_block**(*trigger\_params\_params*)

Set the Trigger Parameters parameters.

#### Parameters

**trigger\_params\_params** (*structs.KMOT\_TriggerParams*) – Options for controlling the trigger.

#### Raises

- **ThorlabsError** – If not successful.
- **TypeError** – If *trigger\_params\_params* is not a *structs.KMOT\_TriggerParams*.

**set\_vel\_params**(*max\_velocity, acceleration*)

Sets the move velocity parameters.

See `get_device_unit_from_real_value()` for converting from a RealValue to a DeviceUnit.

#### Parameters

- **max\_velocity** (`int`) – The maximum velocity in DeviceUnits (see manual).
- **acceleration** (`int`) – The acceleration in DeviceUnits (see manual).

**Raises**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – If not successful.

**stop\_immediate**()

Stop the current move immediately (with the risk of losing track of the position).

**Raises**

*ThorlabsError* – If not successful.

**stop\_polling**()

Stops the internal polling loop.

**stop\_profiled**()

Stop the current move using the current velocity profile.

**Raises**

*ThorlabsError* – If not successful.

**suspend\_move\_messages**()

Suspend automatic messages at ends of moves.

Useful to speed up part of real-time system with lots of short moves.

**Raises**

*ThorlabsError* – 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.
- *bool* – *True* if monitoring is enabled otherwise *False*.



**uses\_pid\_loop\_encoding()**

Determines if we can use PID loop encoding.

This is true if the stage supports PID Loop Encoding. Requires `get_pid_loop_encoder_coeff()` to have a positive non zero coefficient.

**Returns**

`bool` – Whether PID loop encoding is supported.

**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**

`ThorlabsError` – If not successful.

**msl.equipment.resources.thorlabs.kinesis.messages module**

Device Message Queue defined in Thorlabs Kinesis v1.14.18

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.

The message consists of 3 components, a messageType, a messageID and messageData:

```
WORD messageType
WORD messageID
WORD messageData
```

```
msl.equipment.resources.thorlabs.kinesis.messages.MessageTypes = {0:
'GenericDevice', 1: 'GenericPiezo', 2: 'GenericMotor', 3: 'GenericDCMotor',
4: 'GenericSimpleMotor', 5: 'RackDevice', 6: 'Laser', 7: 'TECctlr', 8:
'Quad', 9: 'NanoTrak', 10: 'Specialized', 11: 'Solenoid'}
```

MessageTypes

```
msl.equipment.resources.thorlabs.kinesis.messages.GenericDevice = {0:
'settingsInitialized', 1: 'settingsUpdated', 2: 'settingsExtern', 3:
'error', 4: 'close', 5: 'settingsReset'}
```

GenericDevice

```
msl.equipment.resources.thorlabs.kinesis.messages.GenericMotor = {0: 'Homed',
1: 'Moved', 2: 'Stopped', 3: 'LimitUpdated'}
```

GenericMotor

```
msl.equipment.resources.thorlabs.kinesis.messages.GenericDCMotor = {0:
'error', 1: 'status'}
    GenericDCMotor

msl.equipment.resources.thorlabs.kinesis.messages.GenericPiezo = {0:
'maxVoltageChanged', 1: 'controlModeChanged', 2: 'statusChanged', 3:
'maxTravelChanged', 4: 'TSG_Status', 5: 'TSG_DisplayModeChanged'}
    GenericPiezo

msl.equipment.resources.thorlabs.kinesis.messages.RackDevice = {0:
'RackCountEstablished', 1: 'RackBayState'}
    RackDevice

msl.equipment.resources.thorlabs.kinesis.messages.Quad = {0: 'statusChanged'}
    Quad

msl.equipment.resources.thorlabs.kinesis.messages.TECCtrlr = {0:
'statusChanged', 2: 'displaySettingsChanged', 3: 'feedbackParamsChanged'}
    TECCtrlr

msl.equipment.resources.thorlabs.kinesis.messages.Laser = {0:
'statusChanged', 1: 'controlSourceChanged', 2: 'displayModeChanged'}
    Laser

msl.equipment.resources.thorlabs.kinesis.messages.Solenoid = {0:
'statusChanged'}
    Solenoid

msl.equipment.resources.thorlabs.kinesis.messages.NanoTrak = {0:
'statusChanged'}
    NanoTrak

msl.equipment.resources.thorlabs.kinesis.messages.Specialized = {}
    Specialized

msl.equipment.resources.thorlabs.kinesis.messages.GenericSimpleMotor = {}
    GenericSimpleMotor

msl.equipment.resources.thorlabs.kinesis.messages.MessageID =
{'GenericDCMotor': {0: 'error', 1: 'status'}, 'GenericDevice': {0:
'settingsInitialized', 1: 'settingsUpdated', 2: 'settingsExtern', 3:
'error', 4: 'close', 5: 'settingsReset'}, 'GenericMotor': {0: 'Homed', 1:
'Moved', 2: 'Stopped', 3: 'LimitUpdated'}, 'GenericPiezo': {0:
'maxVoltageChanged', 1: 'controlModeChanged', 2: 'statusChanged', 3:
'maxTravelChanged', 4: 'TSG_Status', 5: 'TSG_DisplayModeChanged'},
'GenericSimpleMotor': {}, 'Laser': {0: 'statusChanged', 1:
'controlSourceChanged', 2: 'displayModeChanged'}, 'NanoTrak': {0:
'statusChanged'}, 'Quad': {0: 'statusChanged'}, 'RackDevice': {0:
'RackCountEstablished', 1: 'RackBayState'}, 'Solenoid': {0:
'statusChanged'}, 'Specialized': {}, 'TECCtrlr': {0: 'statusChanged', 2:
'displaySettingsChanged', 3: 'feedbackParamsChanged'}}
```

MessageID

**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*,  
*build\_device\_list=F*

Bases: `ConnectionSDK`

Base `Thorlabs.MotionControl` class.

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

**Parameters**

- **record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.
- **api\_function** (*api\_functions*) – An API function list from *api\_functions* that the subclass is a wrapper around.
- **build\_device\_list** (*bool*, optional) – Whether to call *build\_device\_list()* before opening the connection to the device.

**Raises**

*ThorlabsError* – 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\_Inertial\_Motor = 97**

KCube Inertial Motor device ID

**KCube\_LaserSource = 56**

KCube Laser Source device ID

**KCube\_NanoTrak = 57**

KCube NanoTrak 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

**Vertical\_Stage = 24**

Vertical Stage 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.

#### **property settings**

The device settings specified in `ThorlabsDefaultSettings.xml`

If this is an empty **dict** then you can specify the `device_name` in the `properties` field in the [Connections Database](#) or you can run the Kinesis software and allow Kinesis to configure the actuator that is connected to the motor controller.

The possible values for `device_name` can be found in the `ThorlabsDefaultSettings.xml` file (located in the Kinesis installation folder, e.g., `C:\Program Files\Thorlabs\Kinesis`), as the `Name` value in one of the `<DeviceSettingsType>` tags.

**Type**

**dict**

**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**

*ThorlabsError* – 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**

*ThorlabsError* – 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**

*ThorlabsError* – 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 `dict`.

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**

`dict` – The message represented as

```
{ 'type': MessageTypes, 'id': MessageID, 'data': int }
```

**msl.equipment.resources.thorlabs.kinesis.structs module**

Structs defined in Thorlabs Kinesis v1.14.10

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TLI_DeviceInfo`

Bases: `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: `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: Structure

**acceleration**

Structure/Union member

**maxVelocity**

Structure/Union member

**minVelocity**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.MOT\_JogParameters

Bases: 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: 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: 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: 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: 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\_BrushlessPositionLoopParameters

Bases: 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_BrushlessTrackSettleParameters
```

Bases: 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_BrushlessCurrentLoopParameters
```

Bases: 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_BrushlessElectricOutputParameters`

Bases: 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: 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: Structure

**movePercentage**

Structure/Union member

**restPercentage**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters

Bases: 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: 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: Structure

**horizontalComponent**

Structure/Union member

**verticalComponent**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.NT\_CircleParameters

Bases: 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: Structure

**LUTdiameter**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.NT\_TIRangeParameters

Bases: 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: 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: 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: 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: Structure

**controlMode**  
Structure/Union member

**gain**  
Structure/Union member

**class**

**msl.equipment.resources.thorlabs.kinesis.structs.PZ\_FeedbackLoopConstants**

Bases: Structure

**integralTerm**

Structure/Union member

**proportionalTerm**

Structure/Union member

**class msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUTWaveParameters**

Bases: 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: Structure

**PIDConstsD**

Structure/Union member

**PIDConstsDFc**

Structure/Union member

**PIDConstsI**

Structure/Union member

**PIDConstsP**

Structure/Union member



**PIDDerivFilter0n**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.PPC\_NotchParams

Bases: Structure

**filter1Fc**

Structure/Union member

**filter1Q**

Structure/Union member

**filter2Fc**

Structure/Union member

**filter2Q**

Structure/Union member

**filterNo**

Structure/Union member

**notchFilter10n**

Structure/Union member

**notchFilter20n**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.PPC\_IOSettings

Bases: 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: 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: Structure

**ADCspeedValue**

Structure/Union member

**digI010perMode**

Structure/Union member

**digI01PulseWidth**

Structure/Union member

**digI01SignalMode**

Structure/Union member

**digI020perMode**

Structure/Union member

**digI02PulseWidth**

Structure/Union member

**digI02SignalMode**

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: 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: Structure

**thresholdDeflection**

Structure/Union member

**velocity**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.MOT\_PotentiometerSteps

Bases: Structure

**potentiometerStepParameters**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_MMIParams

Bases: Structure

**DisplayDimIntensity**

Structure/Union member

**DisplayIntensity**

Structure/Union member

**DisplayTimeout**

Structure/Union member

**PresetPos1**

Structure/Union member

**PresetPos2**

Structure/Union member

**WheelAcceleration**

Structure/Union member

**WheelDirectionSense**

Structure/Union member

**WheelMaxVelocity**

Structure/Union member

**WheelMode**

Structure/Union member

**reserved**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KMOT\_TriggerConfig

Bases: 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: 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.KIM\_DriveOPParameters

Bases: Structure

**class** msl.equipment.resources.thorlabs.kinesis.structs.KIM\_JogParameters

Bases: Structure

**class**

msl.equipment.resources.thorlabs.kinesis.structs.KIM\_LimitSwitchParameters

Bases: Structure

```
class msl.equipment.resources.thorlabs.kinesis.structs.KIM_HomeParameters
    Bases: Structure

class msl.equipment.resources.thorlabs.kinesis.structs.KIM_MMIParameters
    Bases: Structure

class
msl.equipment.resources.thorlabs.kinesis.structs.KIM_MMISignalParameters
    Bases: Structure

class msl.equipment.resources.thorlabs.kinesis.structs.KIM_TriggerIOConfig
    Bases: Structure

class
msl.equipment.resources.thorlabs.kinesis.structs.KIM_TriggerParamsParameters
    Bases: Structure

class msl.equipment.resources.thorlabs.kinesis.structs.KIM_FeedbackSigParams
    Bases: Structure

class msl.equipment.resources.thorlabs.kinesis.structs.KIM_Status
    Bases: Structure

class msl.equipment.resources.thorlabs.kinesis.structs.KLD_MMIParams
    Bases: Structure

    displayIntensity
        Structure/Union member

    reserved
        Structure/Union member

class msl.equipment.resources.thorlabs.kinesis.structs.KLD_TriggerIOParams
    Bases: Structure

    mode1
        Structure/Union member

    mode2
        Structure/Union member

    polarity1
        Structure/Union member

    polarity2
        Structure/Union member

    reserved1
        Structure/Union member

    reserved2
        Structure/Union member

class msl.equipment.resources.thorlabs.kinesis.structs.KLS_MMIParams
    Bases: Structure
```

**displayIntensity**

Structure/Union member

**reserved**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KLS\_TrigIOParams

Bases: Structure

**model**

Structure/Union member

**mode2**

Structure/Union member

**polarity1**

Structure/Union member

**polarity2**

Structure/Union member

**reserved1**

Structure/Union member

**reserved2**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KNA\_TIRangeParameters

Bases: 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.KNA\_TIRReading

Bases: Structure

**absoluteReading**

Structure/Union member

**relativeReading**

Structure/Union member

**selectedRange**

Structure/Union member

**underOrOverRead**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KNA\_IOSettings

Bases: Structure

**highVoltageOutRange**

Structure/Union member

**highVoltageOutputRoute**

Structure/Union member

**lowVoltageOutRange**

Structure/Union member

**lowVoltageOutputRoute**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KNA\_MMIParams

Bases: Structure

**DisplayIntensity**

Structure/Union member

**WheelAdjustRate**

Structure/Union member

**reserved**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KNA\_TriggerConfig

Bases: Structure

**Trigger1Mode**

Structure/Union member

**Trigger1Polarity**

Structure/Union member

**Trigger2Mode**

Structure/Union member

**Trigger2Polarity**

Structure/Union member

**reserved**

Structure/Union member

**unused1**

Structure/Union member

**unused2**

Structure/Union member

**class**

`msl.equipment.resources.thorlabs.kinesis.structs.KNA_FeedbackLoopConstants`

Bases: Structure

**integralTerm**

Structure/Union member

**proportionalTerm**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TPZ_IOSettings`

Bases: Structure

**class** `msl.equipment.resources.thorlabs.kinesis.structs.KPZ_MMIParams`

Bases: 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: 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: 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: Structure

**differentialGain**

Structure/Union member

**integralGain**

Structure/Union member

**proportionalGain**

Structure/Union member

**class**`msl.equipment.resources.thorlabs.kinesis.structs.QD_LowPassFilterParameters`

Bases: Structure

**lowPassFilterCutOffFreq**

Structure/Union member

**lowPassFilterEnabled**

Structure/Union member

**class**

`msl.equipment.resources.thorlabs.kinesis.structs.QD_NotchFilterParameters`

Bases: Structure

**notchFilterCenterFrequency**

Structure/Union member

**notchFilterEnabled**

Structure/Union member

**notchFilterQ**

Structure/Union member

**class**

`msl.equipment.resources.thorlabs.kinesis.structs.QD_PositionDemandParameters`

Bases: 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: Structure

**x**

Structure/Union member

**y**

Structure/Union member

**class** `msl.equipment.resources.thorlabs.kinesis.structs.QD_Readings`

Bases: Structure

**demandedPos**

Structure/Union member

**posDifference**

Structure/Union member

**sum**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.QD\_KPA\_TrigIOConfig

Bases: 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: Structure

**wDigOPs**

Structure/Union member

**wReserved**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.SC\_CycleParameters

Bases: Structure

**closedTime**

Structure/Union member

**numCycles**

Structure/Union member

**openTime**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KSC\_MMIParams

Bases: 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: 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.TSG\_IOSettings

Bases: 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.KSG\_MMIParams

Bases: Structure

**DisplayDimIntensity**

Structure/Union member

**DisplayIntensity**

Structure/Union member

**DisplayTimeout**

Structure/Union member

**reserved**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.KSG\_TriggerConfig

Bases: Structure

**LowerLimit**

Structure/Union member

**SmoothingSamples**

Structure/Union member

**Trigger1Mode**

Structure/Union member

**Trigger1Polarity**

Structure/Union member

**Trigger2Mode**

Structure/Union member

**Trigger2Polarity**

Structure/Union member

**UpperLimit**

Structure/Union member

**reserved**

Structure/Union member

**class** msl.equipment.resources.thorlabs.kinesis.structs.TIM\_DriveOPParameters

Bases: Structure

**class** msl.equipment.resources.thorlabs.kinesis.structs.TIM\_JogParameters

Bases: Structure

**class** msl.equipment.resources.thorlabs.kinesis.structs.TIM\_ButtonParameters

Bases: Structure

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TIM_Status`

Bases: `Structure`

**class** `msl.equipment.resources.thorlabs.kinesis.structs.TC_LoopParameters`

Bases: `Structure`

**differentialGain**

`Structure/Union member`

**integralGain**

`Structure/Union member`

**proportionalGain**

`Structure/Union member`

## Submodules

### `msl.equipment.resources.thorlabs.fwxx2c` module

Wrapper around Thorlabs `FilterWheel102.dll`, v4.0.0.

Thorlabs FW102C Series and FW212C Series Motorized Filter Wheels.

**class** `msl.equipment.resources.thorlabs.fwxx2c.FilterCount`(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: `IntEnum`

The number of filter positions that the filter wheel has.

**SIX** = 6

**TWELVE** = 12

**class** `msl.equipment.resources.thorlabs.fwxx2c.SensorMode`(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: `IntEnum`

Sensor modes of the filter wheel.

**ON** = 0

**OFF** = 1

**class** `msl.equipment.resources.thorlabs.fwxx2c.SpeedMode`(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: `IntEnum`

Speed modes of the filter wheel.

**SLOW** = 0

**FAST** = 1

```
class msl.equipment.resources.thorlabs.fwx2c.TriggerMode(value, names=None, *,
                                                         module=None,
                                                         qualname=None,
                                                         type=None, start=1,
                                                         boundary=None)
```

Bases: `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.fwx2c.FilterWheelXX2C(record)
```

Bases: `ConnectionSDK`

Wrapper around Thorlabs FilterWheel102.dll, v4.0.0.

Connects to the Thorlabs FW102C Series and FW212C Series Motorized Filter Wheels.

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 *properties* for a FilterWheelXX2C connection supports the following key-value pairs in the *Connections Database*:

```
'port': str, the serial port number, e.g., 'COM3'
'baud_rate': int, the baud rate for the serial connection [default: ↵
↵ 115200]
'timeout': int, the timeout in seconds [default: 10]
```

Do not instantiate this class directly. Use the `connect()` method to connect to the equipment.

#### Parameters

**record** (*EquipmentRecord*) – A record from an *Equipment-Register Database*.

#### Raises

**ThorlabsError** – 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* – 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 a 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.

**msl.equipment.utils module**

Common functions.

`msl.equipment.utils.convert_to_enum(obj, enum, prefix=None, to_upper=False, strict=True)`

Convert *obj* to an *Enum* member.

**Parameters**

- **obj** (*object*) – Any object to be converted to the specified *enum*. Can be a value of member of the specified *enum*.
- **enum** (*Type[Enum]*) – The *Enum* object that *obj* should be converted to.

- **prefix** (*str*, optional) – If *obj* is a *str*, then ensures that *prefix* is included at the beginning of *obj* before converting *obj* to the *enum*.
- **to\_upper** (*bool*, optional) – If *obj* is a *str*, then whether to change *obj* to be upper case before converting *obj* to the *enum*.
- **strict** (*bool*, optional) – Whether errors should be raised. If *False* and *obj* cannot be converted to *enum* then *obj* is returned and the error is logged.

**Returns**

*Enum* – The *enum* member.

**Raises**

*ValueError* – If *obj* is not in *enum* and *strict* is *True*.

`msl.equipment.utils.convert_to_primitive(text)`

Convert text into a primitive value.

**Parameters**

**text** (*str* or *bytes*) – The text to convert.

**Returns**

- The *text* as a *None*, *bool*, *int*,
- *float* or *complex* object. Returns the
- original *text* if it cannot be converted to any of these types.
- The *text* *0* and *1* get converted to an integer not a boolean.

`msl.equipment.utils.convert_to_date(obj, fmt='%Y-%m-%d', strict=True)`

Convert an object to a *datetime.date* object.

**Parameters**

- **obj** (*datetime.date*, *datetime.datetime* or *str*) – Any object that can be converted to a *datetime.date* object.
- **fmt** (*str*) – If *obj* is a *str* then the format to use to convert *obj* to a *datetime.date*.
- **strict** (*bool*, optional) – Whether errors should be raised. If *False* and *obj* cannot be converted to *datetime.date* then *datetime.date(datetime.MINYEAR, 1, 1)* is returned and the error is logged.

**Returns**

*datetime.date* – A *datetime.date* object.

`msl.equipment.utils.convert_to_xml_string(element, indent=' ', encoding='utf-8', fix_newlines=True)`

Convert an XML *Element* in to a string with proper indentation.

**Parameters**

- **element** (*Element*) – The element to convert.
- **indent** (*str*, optional) – The value to use for the indentation.
- **encoding** (*str*, optional) – The encoding to use.

- **fix\_newlines** (*bool*, optional) – Whether to remove newlines inside text nodes.

**Returns**

*str* – The *element* as a pretty string. The returned value can be directly written to a file (i.e., it includes the XML declaration).

**Examples**

If the *Element* contains unicode characters then you should use the *codecs* module to create the file if you are using Python 2.7:

```
import codecs
with codecs.open('my_file.xml', mode='w', encoding='utf-8') as fp:
    fp.write(convert_to_xml_string(element))
```

otherwise you can use the builtin *open()* function:

```
with open('my_file.xml', mode='w', encoding='utf-8') as fp:
    fp.write(convert_to_xml_string(element))
```

`msl.equipment.utils.xml_element(tag, text=None, tail=None, **attributes)`

Create a new XML element.

**Parameters**

- **tag** (*str*) – The element's name.
- **text** (*str*, optional) – The text before the first sub-element. Can either be a string or *None*.
- **tail** (*str*, optional) – The text after this element's end tag, but before the next sibling element's start tag.
- **attributes** – All additional key-value pairs are included as XML attributes for the element. The value must be of type *str*.

**Returns**

*Element* – The new XML element.

`msl.equipment.utils.xml_comment(text)`

Create a new XML comment element.

**Parameters**

**text** (*str*) – The comment.

**Returns**

*Comment()* – A special element that is an XML comment.

`msl.equipment.utils.to_bytes(iterable, fmt='ieee', dtype='<f')`

Convert an iterable of numbers into bytes.

**Parameters**

- **iterable** – An object to convert to bytes. Must be a 1-dimensional sequence of elements (not a multidimensional array).

- **fmt** (*str* or *None*, optional) – The format to use to convert *iterable*. Possible values are:

- `' '` (empty string or *None*) – convert *iterable* to bytes without a header.

*None*: <byte><byte><byte>...

- `'ascii'` – comma-separated ASCII characters, see the *<PROGRAM DATA SEPARATOR>* standard that is defined in Section 7.4.2.2, IEEE 488.2-1992.

ascii: <string>,<string>,<string>,...

- `'ieee'` – arbitrary block data for *SCPI* messages, see the *<DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>* standard that is defined in Section 8.7.9, IEEE 488.2-1992.

ieee: #<length of num bytes value><num bytes><byte><byte><byte>...

- `'hp'` – the HP-IB data transfer standard, i.e., the *FORM#* command option. See the programming guide for an HP 8530A for more details.

hp: #A<num bytes as uint16><byte><byte><byte>...

- **dtype** – The data type to use to convert each element in *iterable* to. If *fmt* is `'ascii'` then *dtype* must be of type *str* and it is used as the *format\_spec* argument in `format()` to first convert each element in *iterable* to a string, and then it is encoded (e.g., `'.2e'` converts each element to scientific notation with two digits after the decimal point). If *dtype* includes a byte-order character, it is ignored. For all other values of *fmt*, the *dtype* can be any object that `numpy.dtype` supports (e.g., `'H'`, `'uint16'` and `numpy.ushort` are equivalent values to convert each element to an *unsigned short*). If a byte-order character is specified then it is used, otherwise the native byte order of the CPU architecture is used. See *Format Strings* for more details.

### Returns

*bytes* – The *iterable* converted to bytes.

`msl.equipment.utils.from_bytes(buffer, fmt='ieee', dtype='<f')`

Convert bytes into an array.

### Parameters

- **buffer** (*bytes*, *bytearray* or *str*) – A byte buffer. Can be an already-decoded buffer of type *str*, but only if *fmt* equals `'ascii'`.
- **fmt** (*str* or *None*, optional) – The format that *buffer* is in. See `to_bytes()` for more details.
- **dtype** – The data type of each element in *buffer*. Can be any object that `numpy.dtype` supports. See `to_bytes()` for more details.

### Returns

`numpy.ndarray` – The array.

`msl.equipment.utils.ipv4_addresses()`

Get the IPv4 addresses of the computer.

**Returns**

`set` of `str` – All IPv4 addresses on all network interfaces.

`msl.equipment.utils.parse_lxi_webserver(host, port=80, timeout=1)`

Get the information about an LXI device from the device's webserver.

**Parameters**

- **host** (`str`) – The IP address or hostname of the LXI device.
- **port** (`int`, optional) – The port number of the device's webservice.
- **timeout** (`float`, optional) – The maximum number of seconds to wait for a reply.

**Returns**

`dict` – The information about the LXI device.

## **msl.equipment.vxi11 module**

Implementation of the [VXI-11](#) protocol.

[VXI-11](#) is a client-service model to send messages through a network. The messages are formatted using the Remote Procedure Call protocol [[RFC-1057](#)] and are encoded/decoded using the eXternal Data Representation standard [[RFC-1014](#)].

## **References**

- [VXI-11 – TCP/IP Instrument Protocol Specification \(Revision 1.0\)](#), **VXIbus Consortium**, July 1995.
- [RFC-1057 – RPC: Remote Procedure Call Protocol Specification \(Version 2\)](#), **Sun Microsystems**, June 1988.
- [RFC-1014 – XDR: External Data Representation Standard](#), **Sun Microsystems**, June 1987.

**class** `msl.equipment.vxi11.OperationFlag`(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: `IntEnum`

*VXI-11*: Additional information concerning how a request is carried out.

**NULL** = 0

**WAITLOCK** = 1

**END** = 8

**TERMCHRSET** = 128

```
class msl.equipment.vxi11.MessageType(value, names=None, *, module=None,
                                     qualname=None, type=None, start=1,
                                     boundary=None)
```

Bases: `IntEnum`

*RPC*: The message type.

**CALL** = 0

**REPLY** = 1

```
class msl.equipment.vxi11.ReplyStatus(value, names=None, *, module=None,
                                       qualname=None, type=None, start=1,
                                       boundary=None)
```

Bases: `IntEnum`

*RPC*: Message reply status.

**MSG\_ACCEPTED** = 0

**MSG\_DENIED** = 1

```
class msl.equipment.vxi11.AcceptStatus(value, names=None, *, module=None,
                                       qualname=None, type=None, start=1,
                                       boundary=None)
```

Bases: `IntEnum`

*RPC*: Message accepted status.

**SUCCESS** = 0

**PROG\_UNAVAIL** = 1

**PROG\_MISMATCH** = 2

**PROC\_UNAVAIL** = 3

**GARBAGE\_ARGS** = 4

```
class msl.equipment.vxi11.RejectStatus(value, names=None, *, module=None,
                                       qualname=None, type=None, start=1,
                                       boundary=None)
```

Bases: `IntEnum`

*RPC*: Message rejected status.

**RPC\_MISMATCH** = 0

**AUTH\_ERROR** = 1

```
class msl.equipment.vxi11.AuthStatus(value, names=None, *, module=None,
                                       qualname=None, type=None, start=1,
                                       boundary=None)
```

Bases: `IntEnum`

*RPC*: Authorization status.

`AUTH_BADCRED = 1`

`AUTH_REJECTEDCRED = 2`

`AUTH_BADVERF = 3`

`AUTH_REJECTEDVERF = 4`

`AUTH_TOOWEAK = 5`

**class** `msl.equipment.vxi11.RPCClient`(*host*)

Bases: `object`

Remote Procedure Call implementation for a client.

**Parameters**

**host** (`str`) – The hostname or IP address of the remote device.

**append**(*data*)

Append data to the body of the current RPC message.

**Parameters**

**data** (`bytes` or `memoryview`) – The data to append.

**append\_opaque**(*text*)

Append a variable-length string to the body of the current RPC message.

**Parameters**

**text** (`memoryview`, `bytes` or `str`) – The data to append.

**property chunk\_size**

The maximum number of bytes to receive at a time from the socket.

**Type**

`int`

**close**()

Close the RPC socket, if one is open.

**connect**(*port*, *timeout=10*)

Connect to a specific port on the device.

**Parameters**

- **port** (`int`) – The port number to connect to.
- **timeout** (`float` or `None`, optional) – The maximum number of seconds to wait for the connection to be established.

**get\_buffer**()

Get the data in the buffer.

**Returns**

`bytearray` – The data in the current RPC message.

**get\_port**(*prog*, *vers*, *prot*, *timeout=10*)

Call the Port Mapper procedure to determine which port to use for a program.

This method will automatically open and close the socket connection.



**Parameters**

- **prog** (*int*) – The program number to get the port number of.
- **vers** (*int*) – The version number of *prog*.
- **prot** (*int*) – The socket protocol family type to use when sending requests to *prog* (IPPROTO\_TCP or IPPROTO\_UDP).
- **timeout** (*float* or *None*, optional) – The maximum number of seconds to wait to get the port value.

**Returns**

*int* – The port number that corresponds to *prog*.

**init**(*prog*, *vers*, *proc*)

Construct a new RPC message.

**Parameters**

- **prog** (*int*) – The program number.
- **vers** (*int*) – The version number of program.
- **proc** (*int*) – The procedure number within the program to be called.

**interrupt\_handler**()

Override this method to be notified of a service interrupt.

This method gets called if an interrupt is received during a *read()*. It does not continuously poll the device.

**read**()

Read an RPC message, check for errors, and return the procedure-specific data.

**Returns**

*memoryview* – The procedure-specific data.

**set\_timeout**(*timeout*)

Set the socket timeout value.

**Parameters**

**timeout** (*float*) – The timeout, in seconds, to use for the socket.

**property socket**

The reference to the socket.

**Type**

*socket*

**static unpack\_opaque**(*data*)

Unpack and return a variable-length string.

**Parameters**

**data** (*bytes*, *bytearray* or *memoryview*) – The data to unpack.

**Returns**

*bytes*, *bytearray* or *memoryview* – The unpacked data.

**write()**

Write the RPC message that is in the buffer.

**check\_reply(message)**

Checks the message for errors and returns the procedure-specific data.

**Parameters**

**message** ([memoryview](#)) – The reply from an RPC message.

**Returns**

[memoryview](#) or [None](#) – The reply or [None](#) if the transaction id does not match the value that was used in the corresponding [write\(\)](#) call.

**class msl.equipment.vxi11.VXIClient(host)**

Bases: [RPCClient](#)

Base class for a VXI-11 program.

**Parameters**

**host** ([str](#)) – The hostname or IP address of the remote device.

**read\_reply()**

Check the RPC message for an error and return the remaining data.

**Returns**

[memoryview](#) – The reply data.

**class msl.equipment.vxi11.CoreClient(host)**

Bases: [VXIClient](#)

Communicate with the *Device Core* program on the remote device.

**Parameters**

**host** ([str](#)) – The hostname or IP address of the remote device.

**create\_link(device, lock\_device, lock\_timeout)**

Create a link.

**Parameters**

- **device** ([bytes](#) or [str](#)) – Name of the device to link with.
- **lock\_device** ([bool](#)) – Whether to attempt to lock the device.
- **lock\_timeout** ([int](#)) – Time, in milliseconds, to wait on a lock.

**Returns**

- [int](#) – The link ID.
- [int](#) – The port number of the *Device Async* program (see [AsyncClient](#)).
- [int](#) – The maximum data size the device will accept on a [device\\_write\(\)](#).

**device\_write(lid, io\_timeout, lock\_timeout, flags, data)**

Write data to the specified device.

**Parameters**

- **lid** (*int*) – Link id from `create_link()`.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **data** (*memoryview*, *bytes* or *str*) – The data to write.

**Returns**

*int* – The number of bytes written.

**device\_read**(*lid*, *request\_size*, *io\_timeout*, *lock\_timeout*, *flags*, *term\_char*)

Read data from the device.

**Parameters**

- **lid** (*int*) – Link id from `create_link()`.
- **request\_size** (*int*) – The number of bytes requested.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **term\_char** (*int*) – The termination character. Valid only if *flags* is `TERMCHRSET`.

**Returns**

- *int* – The reason(s) the read completed.
- *memoryview* – A view of the data (the RPC header is removed).

**device\_readstb**(*lid*, *flags*, *lock\_timeout*, *io\_timeout*)

Read the status byte from the device.

**Parameters**

- **lid** (*int*) – Link id from `create_link()`.
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.

**Returns**

*int* – The status byte.

**device\_trigger**(*lid*, *flags*, *lock\_timeout*, *io\_timeout*)

Send a trigger to the device.

**Parameters**

- **lid** (*int*) – Link id from `create_link()`.
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.

**device\_clear**(*lid*, *flags*, *lock\_timeout*, *io\_timeout*)

Send the *clear* command to the device.

**Parameters**

- **lid** (*int*) – Link id from [create\\_link\(\)](#).
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.

**device\_remote**(*lid*, *flags*, *lock\_timeout*, *io\_timeout*)

Place the device in a remote state wherein all programmable local controls are disabled.

**Parameters**

- **lid** (*int*) – Link id from [create\\_link\(\)](#).
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.

**device\_local**(*lid*, *flags*, *lock\_timeout*, *io\_timeout*)

Place the device in a local state wherein all programmable local controls are enabled.

**Parameters**

- **lid** (*int*) – Link id from [create\\_link\(\)](#).
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.

**device\_lock**(*lid*, *flags*, *lock\_timeout*)

Acquire a device's lock.

**Parameters**

- **lid** (*int*) – Link id from [create\\_link\(\)](#).
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.

**device\_unlock**(*lid*)

Release a lock acquired by [device\\_lock\(\)](#).

**Parameters**

- **lid** (*int*) – Link id from [create\\_link\(\)](#).

**device\_enable\_srq**(*lid*, *enable*, *handle*)

Enable or disable the sending of *device\_intr\_srq* RPCs by the network instrument server.

**Parameters**

- **lid** (*int*) – Link id from [create\\_link\(\)](#).
- **enable** (*bool*) – Whether to enable or disable interrupts.

- **handle** (*bytes*) – Host specific data (maximum length is 40 characters).

**device\_docmd**(*lid, flags, io\_timeout, lock\_timeout, cmd, network\_order, datasize, data\_in*)

Allows for a variety of operations to be executed.

#### Parameters

- **lid** (*int*) – Link id from [create\\_link\(\)](#).
- **flags** (*int* or *OperationFlag*) – Operation flags to use.
- **io\_timeout** (*int*) – Time, in milliseconds, to wait for I/O to complete.
- **lock\_timeout** (*int*) – Time, in milliseconds, to wait on a lock.
- **cmd** (*int*) – Which command to execute.
- **network\_order** (*bool*) – Client's byte order.
- **datasize** (*int*) – Size of individual data elements.
- **data\_in** (*bytes* or *str*) – Data input parameters.

#### Returns

*bytes* – The results defined by *cmd*.

**destroy\_link**(*lid*)

Destroy the link.

#### Parameters

**lid** (*int*) – Link id from [create\\_link\(\)](#).

**create\_intr\_chan**(*host\_addr, host\_port, prog\_num, prog\_vers, prog\_family*)

Inform the network instrument server to establish an interrupt channel.

#### Parameters

- **host\_addr** (*int*) – Host servicing the interrupt.
- **host\_port** (*int*) – Valid port number on the client.
- **prog\_num** (*int*) – Program number.
- **prog\_vers** (*int*) – Program version number.
- **prog\_family** (*int*) – The underlying socket protocol family type (IPPROTO\_TCP or IPPROTO\_UDP).

**destroy\_intr\_chan**()

Inform the network instrument server to close its interrupt channel.

**class** `msl.equipment.vxi11.AsyncClient`(*host*)

Bases: [VXIClient](#)

Communicate with the *Device Async* program on the remote device.

#### Parameters

**host** (*str*) – The hostname or IP address of the remote device.

**device\_abort**(*lid*)

Stops an in-progress call.

**Parameters**

**lid** (*int*) – Link id from `create_link()`.

`msl.equipment.vxi11.find_vxi11(hosts=None, timeout=1)`

Find all VXI-11 devices that are on the network.

The RPC port-mapper protocol ([RFC-1057](#), Appendix A) broadcasts a message via UDP to port 111 for VXI-11 device discovery.

**Parameters**

- **hosts** (*list* of *str*, optional) – The IP address(es) on the computer to use to broadcast the message. If not specified, then broadcast on all network interfaces.
- **timeout** (*float*, optional) – The maximum number of seconds to wait for a reply.

**Returns**

*dict* – The information about the VXI-11 devices that were found.

## 1.7 License

**MIT License**

Copyright (c) 2017 - 2023, Measurement Standards Laboratory of New Zealand

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.

## 1.8 Developers

- Joseph Borbely <joseph.borbely@measurement.govt.nz>

## 1.9 Release Notes

### 1.9.1 Version 0.1.0 (2023-06-18)

Initial release.

It is also the last release to support Python 2.7, 3.5, 3.6 and 3.7





---

**CHAPTER  
TWO**

---

**INDEX**

- `modindex`



## PYTHON MODULE INDEX

### m

`msl.equipment`, 17

`msl.equipment.config`, 17

`msl.equipment.connection`, 20

`msl.equipment.connection_demo`, 22

`msl.equipment.connection_message_based`,  
22

`msl.equipment.connection_nidaq`, 25

`msl.equipment.connection_prologix`, 27

`msl.equipment.connection_pyvisa`, 31

`msl.equipment.connection_sdk`, 32

`msl.equipment.connection_serial`, 33

`msl.equipment.connection_socket`, 35

`msl.equipment.connection_tcpip_hislip`,  
36

`msl.equipment.connection_tcpip_vxi11`,  
39

`msl.equipment.connection_zeromq`, 42

`msl.equipment.constants`, 43

`msl.equipment.database`, 45

`msl.equipment.dns_service_discovery`, 47

`msl.equipment.exceptions`, 48

`msl.equipment.factory`, 50

`msl.equipment.hislip`, 50

`msl.equipment.resources`, 81

`msl.equipment.resources.aim_tti`, 84

`msl.equipment.resources.aim_tti.mx_series`,  
84

`msl.equipment.resources.avantes`, 91

`msl.equipment.resources.avantes.avaspec`,  
91

`msl.equipment.resources.bentham`, 121

`msl.equipment.resources.bentham.benhw32`,  
122

`msl.equipment.resources.bentham.benhw64`,  
124

`msl.equipment.resources.bentham.errors`,  
125

`msl.equipment.resources.bentham.tokens`,  
125

`msl.equipment.resources.cmi`, 125

`msl.equipment.resources.cmi.sia3`, 125

`msl.equipment.resources.dataray`, 127

`msl.equipment.resources.dataray.datarayocx_32`,  
127

`msl.equipment.resources.dataray.datarayocx_64`,  
127

`msl.equipment.resources.dmm`, 81

`msl.equipment.resources.electron_dynamics`,  
130

`msl.equipment.resources.electron_dynamics.tc_series`,  
130

`msl.equipment.resources.energetiq`, 135

`msl.equipment.resources.energetiq.eq99`,  
135

`msl.equipment.resources.mks_instruments`,  
141

`msl.equipment.resources.mks_instruments.pr4000b`,  
141

`msl.equipment.resources.nkt`, 153

`msl.equipment.resources.nkt.nktpdll`,  
153

`msl.equipment.resources.omega`, 186

`msl.equipment.resources.omega.ithx`, 186

`msl.equipment.resources.optosigma`, 190

`msl.equipment.resources.optosigma.shot702`,  
190

`msl.equipment.resources.optronic_laboratories`,  
197

`msl.equipment.resources.optronic_laboratories.ol756`,  
197

`msl.equipment.resources.optronic_laboratories.ol756`,  
215

`msl.equipment.resources.optronic_laboratories.ol_cu`,  
216

`msl.equipment.resources.picotech`, 219

`msl.equipment.resources.picotech.errors`,  
219

`msl.equipment.resources.picotech.picoscope`,  
220

`msl.equipment.resources.picotech.picoscope.callback`,  
220

`msl.equipment.resources.picotech.picoscope.channels,`  
    [222](#) `msl.equipment.resources.thorlabs.kinesis.callbacks,`  
`msl.equipment.resources.picotech.picoscope.enums,`  
    [224](#) `msl.equipment.resources.thorlabs.kinesis.enums,`  
`msl.equipment.resources.picotech.picoscope.functions,`  
    [310](#) `msl.equipment.resources.thorlabs.kinesis.errors,`  
`msl.equipment.resources.picotech.picoscope.helpers,`  
    [310](#) `msl.equipment.resources.thorlabs.kinesis.filter_flow,`  
`msl.equipment.resources.picotech.picoscope.picoscope,`  
    [312](#) `msl.equipment.resources.thorlabs.kinesis.integrated,`  
`msl.equipment.resources.picotech.picoscope.picoscope_2k3k,`  
    [316](#) `msl.equipment.resources.thorlabs.kinesis.kcube_dc_s,`  
`msl.equipment.resources.picotech.picoscope.picoscope_api,`  
    [318](#) `msl.equipment.resources.thorlabs.kinesis.kcube_sole,`  
`msl.equipment.resources.picotech.picoscope.ps2000,`  
    [329](#) `msl.equipment.resources.thorlabs.kinesis.kcube_step,`  
`msl.equipment.resources.picotech.picoscope.ps2000a,`  
    [331](#) `msl.equipment.resources.thorlabs.kinesis.messages,`  
`msl.equipment.resources.picotech.picoscope.ps3000,`  
    [332](#) `msl.equipment.resources.thorlabs.kinesis.motion_con,`  
`msl.equipment.resources.picotech.picoscope.ps3000a,`  
    [334](#) `msl.equipment.resources.thorlabs.kinesis.structs,`  
`msl.equipment.resources.picotech.picoscope.ps4000,`  
    [336](#) `msl.equipment.resources.utils,` [82](#)  
`msl.equipment.resources.picotech.picoscope.ps4000a,` `msl.equipment.utils,` [582](#)  
    [338](#) `msl.equipment.vxi11,` [586](#)  
`msl.equipment.resources.picotech.picoscope.ps5000,`  
    [340](#)  
`msl.equipment.resources.picotech.picoscope.ps5000a,`  
    [341](#)  
`msl.equipment.resources.picotech.picoscope.ps6000,`  
    [343](#)  
`msl.equipment.resources.picotech.picoscope.structs,`  
    [345](#)  
`msl.equipment.resources.picotech.pt104,`  
    [357](#)  
`msl.equipment.resources.princeton_instruments,`  
    [361](#)  
`msl.equipment.resources.princeton_instruments.arc_instrument,`  
    [361](#)  
`msl.equipment.resources.raicol,` [388](#)  
`msl.equipment.resources.raicol.raicol_tec,`  
    [388](#)  
`msl.equipment.resources.thorlabs,` [389](#)  
`msl.equipment.resources.thorlabs.fwxx2c,`  
    [578](#)  
`msl.equipment.resources.thorlabs.kinesis,`  
    [389](#)  
`msl.equipment.resources.thorlabs.kinesis.api_functions,`  
    [389](#)  
`msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor,`



absoluteReading (attribute), 261  
 (msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Bunch), 556  
 attribute), 563 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261

AC (msl.equipment.resources.picotech.picoscope.enums.PS2000ACCoupling), 261  
 attribute), 235 ACCELEROMETER\_50V  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS3000ACCoupling), 261  
 attribute), 251 (msl.equipment.resources.picotech.picoscope.enums.PS4000ACCoupling), 261

AC (msl.equipment.resources.picotech.picoscope.enums.PS4000ACCoupling), 261  
 attribute), 270 (msl.equipment.resources.picotech.picoscope.enums.PS4000ACCoupling), 261

AC (msl.equipment.resources.picotech.picoscope.enums.PS5000ACCoupling), 261  
 attribute), 293 AcceptStatus (class in msl.equipment.vxi11),  
 AC (msl.equipment.resources.picotech.picoscope.enums.PS6000ACCoupling), 304  
 attribute), 304 accumulate\_signals()  
 ACCELERATION (msl.equipment.resources.thorlabs.kinesis.enums.LegacyType), 462  
 attribute), 462 (msl.equipment.resources.optronic\_laboratories.ol756ocx), 197  
 acceleration (msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Bunch), 556  
 attribute), 556 activate() (msl.equipment.resources.avantes.avaspec.Avantes), 110  
 method), 110 accelerationFeedForward  
 ADCspeedValue (msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Bunch), 556  
 (msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Bunch), 556  
 attribute), 558 ADD (msl.equipment.resources.picotech.picoscope.enums.PS4000ACChannelInfo), 261  
 ACCELEROMETER (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelInfo), 262  
 attribute), 262 address (msl.equipment.record\_types.ConnectionRecord), 79  
 ACCELEROMETER\_100MV  
 attribute), 79  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 229  
 attribute), 261 ADV\_FINAL (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 229  
 ACCELEROMETER\_100V  
 ADV\_NONE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 229  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 261 ADV\_RISING (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 229  
 ACCELEROMETER\_10MV  
 attribute), 229  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 261 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 241  
 ACCELEROMETER\_10V  
 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 258  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 261 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 280  
 ACCELEROMETER\_1V  
 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 268  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 268 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 291  
 ACCELEROMETER\_200MV  
 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 309  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 261 AGGREGATE (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 309  
 ACCELEROMETER\_20MV  
 attribute), 309  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 261 AimTTiError, 48  
 ACCELEROMETER\_20V  
 aimonist (msl.equipment.resources.thorlabs.kinesis.structs.NT\_Circuit), 562  
 attribute), 261 (msl.equipment.resources.thorlabs.kinesis.structs.NT\_Circuit), 562  
 ACCELEROMETER\_2V  
 alias (msl.equipment.record\_types.EquipmentRecord), 73  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 attribute), 261 allocate() (msl.equipment.resources.picotech.picoscope.channel), 261  
 ACCELEROMETER\_500MV  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000Range), 261





|   |   |  |
|---|---|--|
| <i>attribute</i> ), 51  | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                      | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51     |
| AsyncInterrupted  | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                      | AsyncStatusQuery (class in <i>msl.equipment.hislip</i> ), 62 |
| AsyncLock (class in <i>msl.equipment.hislip</i> ), 56                       | AsyncStatusQuery  | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51     |
| AsyncLock ( <i>msl.equipment.hislip.MessageType attribute</i> ), 50         | AsyncStatusResponse (class in <i>msl.equipment.hislip</i> ), 62               |  |
| AsyncLockInfo (class in <i>msl.equipment.hislip</i> ), 57                   | AsyncStatusResponse   | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51     |
| AsyncLockInfo ( <i>msl.equipment.hislip.MessageType attribute</i> ), 51     | attrib() ( <i>msl.equipment.config.Config</i> method), 19                     |  |
| AsyncLockInfoResponse (class in <i>msl.equipment.hislip</i> ), 57           | AUTH_BADCRED ( <i>msl.equipment.vxi11.AuthStatus attribute</i> ), 587         |  |
| AsyncLockInfoResponse   | AUTH_BADVERF ( <i>msl.equipment.vxi11.AuthStatus attribute</i> ), 588         |  |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                    | AUTH_ERROR ( <i>msl.equipment.vxi11.RejectStatus attribute</i> ), 587         |  |
| AsyncLockResponse (class in <i>msl.equipment.hislip</i> ), 56               | AUTH_REJECTEDCRED   | <i>(msl.equipment.vxi11.AuthStatus attribute)</i> , 588      |
| AsyncLockResponse   | AUTH_REJECTEDVERF   | <i>(msl.equipment.vxi11.AuthStatus attribute)</i> , 588      |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 50                    | AUTH_TOOWEAK ( <i>msl.equipment.vxi11.AuthStatus attribute</i> ), 588         |  |
| AsyncMaximumMessageSize (class in <i>msl.equipment.hislip</i> ), 60         | AUTHENTICATION_FAILED   | <i>(msl.equipment.hislip.ErrorType attribute)</i> , 52       |
| AsyncMaximumMessageSize   | authentication_result() ( <i>msl.equipment.hislip.SyncClient method</i> ), 70 |  |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                    | authentication_start() ( <i>msl.equipment.hislip.SyncClient method</i> ), 70  |  |
| AsyncMaximumMessageSizeResponse (class in <i>msl.equipment.hislip</i> ), 60 | AuthenticationExchange (class in <i>msl.equipment.hislip</i> ), 66            |  |
| AsyncMaximumMessageSizeResponse   | AuthenticationExchange  | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51     |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                    | AuthenticationResult (class in <i>msl.equipment.hislip</i> ), 66              |  |
| AsyncRemoteLocalControl (class in <i>msl.equipment.hislip</i> ), 58         | AuthenticationResult  | <i>(msl.equipment.hislip.MessageType attribute)</i> , 52     |
| AsyncRemoteLocalControl   | AuthenticationStart (class in <i>msl.equipment.hislip</i> ), 65               |  |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                    | AuthenticationStart   | <i>(msl.equipment.hislip.MessageType attribute)</i> , 51     |
| AsyncRemoteLocalResponse (class in <i>msl.equipment.hislip</i> ), 58        |   |  |
| AsyncRemoteLocalResponse  |   |  |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                    |   |  |
| AsyncServiceRequest   |   |  |
| <i>(msl.equipment.hislip.MessageType attribute)</i> , 51                    |   |  |
| AsyncStartTLS (class in <i>msl.equipment.hislip</i> ), 63                   |   |  |
| AsyncStartTLS ( <i>msl.equipment.hislip.MessageType attribute</i> ), 51     |   |  |
| AsyncStartTLSResponse (class in <i>msl.equipment.hislip</i> ), 63           |   |  |
| AsyncStartTLSResponse   |   |  |



MSL-Equipment Documentation, Release 0.1.0

[mssl.equipment.resources.avantes.avaspec](#)), [attribute](#)), 258  
[102](#) [AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS4000A\\_RatioMode](#)  
[Avantes.IrradianceType](#) (class in [attribute](#)), 280  
[mssl.equipment.resources.avantes.avaspec](#)), [AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS4000A\\_RatioMode](#)  
[106](#) [attribute](#)), 268  
[Avantes.MeasConfigType](#) (class in [AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS5000A\\_RatioMode](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [attribute](#)), 301  
[106](#) [AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS5000A\\_RatioMode](#)  
[Avantes.OemDataType](#) (class in [attribute](#)), 291  
[mssl.equipment.resources.avantes.avaspec](#)), [AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS6000A\\_RatioMode](#)  
[109](#) [attribute](#)), 310  
[Avantes.ProcessControlType](#) (class in [AVS\\_SERIAL\\_LEN](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [\(mssl.equipment.resources.avantes.avaspec.Avantec](#)  
[108](#) [attribute](#)), 100  
[Avantes.SensType](#) (class in [AvsIdentityType](#) (class in  
[mssl.equipment.resources.avantes.avaspec](#)), [mssl.equipment.resources.avantes.avaspec](#)),  
[102](#) [93](#)  
[Avantes.SmoothingType](#) (class in [AWG\\_DAC\\_FREQUENCY](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [\(mssl.equipment.resources.picotech.picoscope.ps4000a.Pico](#)  
[105](#) [attribute](#)), 339  
[Avantes.SpectrumCalibrationType](#) (class in [AWG\\_DAC\\_FREQUENCY](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [\(mssl.equipment.resources.picotech.picoscope.ps5000a.Pico](#)  
[106](#) [attribute](#)), 342  
[Avantes.SpectrumCorrectionType](#) (class in [AWG\\_PHASE\\_ACCUMULATOR](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [\(mssl.equipment.resources.picotech.picoscope.ps4000a.Pico](#)  
[106](#) [attribute](#)), 339  
[Avantes.StandAloneType](#) (class in [AWG\\_PHASE\\_ACCUMULATOR](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [\(mssl.equipment.resources.picotech.picoscope.ps5000a.Pico](#)  
[107](#) [attribute](#)), 342  
[Avantes.TecControlType](#) (class in [axisID \(mssl.equipment.resources.thorlabs.kinesis.structs.MOT\\_Sta](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [attribute](#)), 557  
[108](#)  
**B**  
[Avantes.TempSensorType](#) (class in [B \(mssl.equipment.resources.picotech.picoscope.enums.PS2000ACha](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [attribute](#)), 231  
[108](#)  
[Avantes.TimeStampType](#) (class in [B \(mssl.equipment.resources.picotech.picoscope.enums.PS2000Cha](#)  
[mssl.equipment.resources.avantes.avaspec](#)), [attribute](#)), 224  
[107](#) [B \(mssl.equipment.resources.picotech.picoscope.enums.PS3000ACha](#)  
[Avantes.TriggerType](#) (class in [attribute](#)), 248  
[mssl.equipment.resources.avantes.avaspec](#)), [B \(mssl.equipment.resources.picotech.picoscope.enums.PS3000Cha](#)  
[106](#) [attribute](#)), 242  
[AvantesError](#), 48 [B \(mssl.equipment.resources.picotech.picoscope.enums.PS4000ACha](#)  
[AVASPEC\\_ERROR\\_MSG\\_LEN](#) [attribute](#)), 270  
[\(mssl.equipment.resources.avantes.avaspec.Avantec](#), [B \(mssl.equipment.resources.picotech.picoscope.enums.PS4000Cha](#)  
[attribute](#)), 100 [attribute](#)), 259  
[AVASPEC\\_MIN\\_MSG\\_LEN](#) [B \(mssl.equipment.resources.picotech.picoscope.enums.PS5000ACha](#)  
[\(mssl.equipment.resources.avantes.avaspec.Avantec](#) [attribute](#)), 294  
[attribute](#)), 100 [B \(mssl.equipment.resources.picotech.picoscope.enums.PS5000Cha](#)  
[AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS1000A\\_RatioMode](#)  
[attribute](#)), 241 [B \(mssl.equipment.resources.picotech.picoscope.enums.PS6000Cha](#)  
[AVERAGE \(mssl.equipment.resources.picotech.picoscope.enums.PS3000A\\_RatioMode](#)

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 231      **baud\_rate** (*msl.equipment.connection\_serial.ConnectionSerial* attribute), 230

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 230

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 271      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 230

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 259      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex* attribute), 294      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex* attribute), 285      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 271

**B\_MAX** (*msl.equipment.resources.picotech.picoscope.enums.PS6000AChannelBufferIndex* attribute), 303      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 271

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 231      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex* attribute), 294

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex* attribute), 294

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 271      **BELOW** (*msl.equipment.resources.picotech.picoscope.enums.PS6000AChannelBufferIndex* attribute), 303

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 259      **BELOW\_LOWER** (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 231

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex* attribute), 294      **BELOW\_LOWER** (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS5000AChannelBufferIndex* attribute), 285      **BELOW\_LOWER** (*msl.equipment.resources.picotech.picoscope.enums.PS4000AChannelBufferIndex* attribute), 271

**B\_MIN** (*msl.equipment.resources.picotech.picoscope.enums.PS6000AChannelBufferIndex* attribute), 303      **BELOW\_LOWER** (*msl.equipment.resources.picotech.picoscope.enums.PS6000AChannelBufferIndex* attribute), 303

**Backend** (*class in msl.equipment.constants*), 43      **attribute**), 267

**backend** (*msl.equipment.record\_types.ConnectionRecord* attribute), 79      **BELOW\_LOWER** (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 231

**BAD\_CONTROL\_CODE**      **BELOW\_LOWER** (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248

    (*msl.equipment.hislip.ErrorType* attribute), 52      **attribute**), 308

**BAD\_HEADER** (*msl.equipment.hislip.ErrorType* attribute), 52      **Benchtop\_Brushless\_Motor** (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Brushless\_Motor* attribute), 551

**BAD\_MESSAGE\_TYPE**      **Benchtop\_NanoTrak** (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_NanoTrak* attribute), 551

    (*msl.equipment.hislip.ErrorType* attribute), 52      **attribute**), 551

**BAD\_VENDOR** (*msl.equipment.hislip.ErrorType* attribute), 52      **Benchtop\_Piezo\_1\_Channel** (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Piezo\_1\_Channel* attribute), 551

**bandwidth** (*msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel* property), 223      **Benchtop\_Piezo\_3\_Channel** (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Piezo\_3\_Channel* attribute), 551

**BATCH\_AND\_SERIAL**      (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Piezo\_3\_Channel* attribute), 551

    (*msl.equipment.resources.picotech.picoscope.enums.PicoScopeBatchAndSerial* attribute), 224      **Benchtop\_Stepper\_Motor\_1\_Channel** (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Stepper\_Motor\_1\_Channel* attribute), 551

**BATCH\_AND\_SERIAL**      (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Stepper\_Motor\_1\_Channel* attribute), 551

    (*msl.equipment.resources.picotech.picoscope.enums.PS2000AChannelBufferIndex* attribute), 226      **Benchtop\_Stepper\_Motor\_3\_Channel** (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Stepper\_Motor\_3\_Channel* attribute), 551

**BATCH\_AND\_SERIAL**      (*msl.equipment.resources.thorlabs.kinesis.motion\_control.Benchtop\_Stepper\_Motor\_3\_Channel* attribute), 551

    (*msl.equipment.resources.picotech.picoscope.enums.PS3000AChannelBufferIndex* attribute), 248

**BenchtopStepperMotor** (class in `busy (msl.equipment.hislip.AsyncStartTLSResponse msl.equipment.resources.thorlabs.kinesis.benchtop_stepper.py)`, 389)  
**buttonMode** (`msl.equipment.resources.thorlabs.kinesis.structs.MO`), 389  
**Bentham** (class in `attribute`), 566  
**Bentham64BW\_20KHZ** (`msl.equipment.resources.picotech.picoscope.enums.PS4`), 124  
**Bentham32** (class in `BW_20MHZ (msl.equipment.resources.picotech.picoscope.enums.PS`), `attribute`), 248  
**BenthamError**, 48  
**BlockReady** (in `module BW_20MHZ (msl.equipment.resources.picotech.picoscope.enums.PS`), `attribute`), 302  
**BNCTriggerOrLowVoltageOut** (`attribute`), 302  
**BNC\_TRIGGER\_MODE\_SETTINGS** (`msl.equipment.resources.thorlabs.kinesis.structs.BNC_TRIGGER_MODE_SETTINGS`), 561  
**BNT\_BNCTriggerModes** (class in `BW_FULL (msl.equipment.resources.picotech.picoscope.enums.PS40`), `attribute`), 270  
**BNT\_CurrentLimit** (class in `attribute`), 293  
**BNT\_FeedbackSignalSelection** (class in `byte_buffer (msl.equipment.connection_socket.ConnectionSocket`), `property`), 35  
**BNT\_IO\_Settings** (class in `property`), 40  
**BNT\_OutputLowPassFilter** (class in `C (msl.equipment.resources.picotech.picoscope.enums.PS2000ACha`), `attribute`), 231  
**Bright** (`msl.equipment.resources.thorlabs.kinesis.enums.PPCutDensity`), 436  
**BroadcastAnswerType** (class in `attribute`), 248  
**buffer** (`msl.equipment.resources.picotech.picoscope.enums.PS4000ACha`), 223  
**build\_device\_list()** (`C (msl.equipment.resources.picotech.picoscope.enums.PS4000Cha`), `static method`), 553  
**build\_group()** (`msl.equipment.resources.bentham.benhw32.Bentham`), 3294  
**build\_system\_model()** (`msl.equipment.resources.bentham.benhw32.Bentham`), 122  
**build\_system\_model()** (`C_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000`), `attribute`), 231  
**busy** (`msl.equipment.hislip.AsyncEndTLSResponse`), 64



attribute), 271

C\_MAX (msl.equipment.resources.picotech.picoscope.camera.PS4000ChannelBufferIndex attribute), 259

C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ACChannelBufferIndex attribute), 294

C\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 285

C\_MAX (msl.equipment.resources.picotech.picoscope.camera.PS6000ChannelBufferIndex attribute), 303

C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS2000ACChannelBufferIndex attribute), 231

C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS3000ChannelBufferIndex attribute), 248

C\_MIN (msl.equipment.resources.picotech.picoscope.camera.PS4000ChannelBufferIndex attribute), 271

C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelBufferIndex attribute), 259

C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000ACChannelBufferIndex attribute), 294

C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS5000ChannelBufferIndex attribute), 286

C\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS6000ChannelBufferIndex attribute), 303

Cage\_Rotator (msl.equipment.resources.thorlabs.kinesis.motioncontrol.MotionControl attribute), 552

CAL\_DATE (msl.equipment.resources.picotech.picoscope.enums.PS3000AcquisitionFormat attribute), 224

CAL\_DATE (msl.equipment.resources.picotech.picoscope.camera.PS2000Info attribute), 226

CAL\_DATE (msl.equipment.resources.picotech.picoscope.enums.PS3000Info attribute), 244

calc\_mono\_slit\_bandpass() (msl.equipment.resources.princeton\_instruments.arc\_mechanics.PrincetonInstruments method), 364

calibration (msl.equipment.record\_types.MeasurementRecord attribute), 77

calibration\_cycle (msl.equipment.record\_types.CalibrationRecord attribute), 76

calibration\_date (msl.equipment.record\_types.CalibrationRecord attribute), 76

CalibrationRecord (class in category (msl.equipment.record\_types.EquipmentRecord attribute), 75

calibrations (msl.equipment.record\_types.EquipmentRecord attribute), 73

CALL (msl.equipment.vxi11.MessageType attribute), 587

callbacks() (msl.equipment.resources.utils.CHeader method), 83

camelcase\_to\_underscore() (in module changeToOddOrEven msl.equipment.resources.utils), 82

can\_device\_lock\_front\_panel() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 123

can\_home() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 524

can\_home() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper method), 468

can\_home() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo method), 490

can\_home() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper method), 524

can\_move\_without\_homing\_first() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 524

can\_move\_without\_homing\_first() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper method), 468

can\_move\_without\_homing\_first() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo method), 490

can\_move\_without\_homing\_first() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper method), 524

capture() (msl.equipment.resources.dataray.datarayocx\_32.DataArray method), 127

capture() (msl.equipment.resources.dataray.datarayocx\_64.DataArray method), 128

change\_power\_source() (msl.equipment.resources.picotech.picoscope.picoscope\_a method), 318

changeToOddOrEven (msl.equipment.resources.thorlabs.kinesis.structs.KNA\_TT attribute), 570

([msl.equipment.resources.thorlabs.kinesis.structs.NTaffRangeParameters](#)  
attribute), 562  
[Channel3](#) ([msl.equipment.resources.thorlabs.kinesis.enums.TIM\\_C](#)  
[channel](#) ([msl.equipment.resources.picotech.picoscope.channels.PicoScopeChannel](#)  
property), 222  
[CHANNEL](#) ([msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_P142StringValue](#)  
attribute), 282  
[channel](#) ([msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScope](#)  
property), 312  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS2600A\\_DigitalChannelDirections](#)  
attribute), 347  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS2600A\\_TriggerChannelProperties](#)  
attribute), 347  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS2600A\\_TriggerChannelProperties](#)  
attribute), 345  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS2600A\\_DigitalChannelDirections](#)  
attribute), 350  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS3600A\\_TriggerChannelProperties](#)  
attribute), 350  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS3600A\\_TriggerChannelProperties](#)  
attribute), 347  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4600A\\_ChannelLedSetting](#)  
attribute), 352  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4600A\\_ConnectDetect](#)  
attribute), 353  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4600A\\_Direction](#)  
attribute), 352  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4600A\\_TriggerChannelProperties](#)  
attribute), 353  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4600A\\_TriggerChannelProperties](#)  
attribute), 352  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS5600A\\_TriggerChannelProperties](#)  
attribute), 356  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS5600A\\_TriggerChannelProperties](#)  
attribute), 354  
[channel](#) ([msl.equipment.resources.picotech.picoscope.structs.PS6600A\\_TriggerChannelProperties](#)  
attribute), 357  
[channel](#) ([msl.equipment.resources.thorlabs.kinesis.structs.BNTriO\\_Settings](#)  
attribute), 561  
[Channel1](#) ([msl.equipment.resources.thorlabs.kinesis.enums.KIMri6Line\\_n2f0](#)  
attribute), 439  
[Channel1](#) ([msl.equipment.resources.thorlabs.kinesis.enums.THMri6Line\\_n2B3](#)  
attribute), 459  
[Channel1Only](#) ([msl.equipment.resources.thorlabs.kinesis.enums.unts.Chary\\_n2E1EnableModes](#)  
attribute), 442  
[Channel2](#) ([msl.equipment.resources.thorlabs.kinesis.enums.KIMri6Line\\_n2f0](#)  
attribute), 439  
[Channel2](#) ([msl.equipment.resources.thorlabs.kinesis.enums.THMri6Line\\_n2E0](#)  
attribute), 459  
[Channel2Only](#) ([msl.equipment.resources.thorlabs.kinesis.enums.unts.Chary\\_n2E1EnableModes](#)  
attribute), 442  
[Channel3](#) ([msl.equipment.resources.thorlabs.kinesis.enums.KIMri6Line\\_n2f0](#)











`clear_message_queue()` (attribute), 78  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoideDCServo (msl.equipment.resources.picotech.picoscope.enums.PS2000Error), 491  
 attribute), 227  
`clear_message_queue()` COMPLETE (msl.equipment.resources.picotech.picoscope.enums.PS2000Error), 491  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoideDCServo (msl.equipment.resources.picotech.picoscope.enums.PS2000Error), 491  
 method), 515  
 component\_select\_wl()  
`clear_message_queue()` (msl.equipment.resources.bentham.benhw32.Bentham32CubeStepperMotor), 524  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_steppermotor2KCubeStepperMotor), 524  
 method), 524  
 condition (msl.equipment.resources.picotech.picoscope.structs.PicoscopeCondition), 524  
`clockwiseHardwareLimit` attribute), 353  
 (msl.equipment.resources.thorlabs.kinesis.structs.KinesisHardwareLimitParameters), 560  
 attribute), 560  
 (msl.equipment.resources.energetiq.eq99.EQ99), 560  
`clockwisePosition` method), 136  
 (msl.equipment.resources.thorlabs.kinesis.structs.KinesisPosition), 560  
 attribute), 560  
 (msl.equipment.resources.thorlabs.kinesis.structs.KinesisPositionParameters), 560  
 attribute), 77  
`close()` (msl.equipment.hislip.HiSLIPClient Config (class in msl.equipment.config), 17  
 method), 67  
 CONFIG\_FAIL (msl.equipment.resources.picotech.errors.PS2000Error), 219  
`close()` (msl.equipment.resources.bentham.benhw32.Bentham32CubeStepperMotor), 122  
 method), 122  
 CONFIG\_FAIL (msl.equipment.resources.picotech.errors.PS3000Error), 220  
`close()` (msl.equipment.resources.thorlabs.fwx2c.FilterWheelV2C), 220  
 method), 579  
 connect() (in module msl.equipment.factory), 50  
`close()` (msl.equipment.resources.thorlabs.kinesis.benchtopStepperMotor), 390  
 connect() (msl.equipment.hislip.HiSLIPClient), 67  
 method), 67  
`close()` (msl.equipment.resources.thorlabs.kinesis.filmStripper), 463  
 from\_type() (msl.equipment.record\_types.EquipmentRecord), 74  
 method), 74  
`close()` (msl.equipment.resources.thorlabs.kinesis.independentStepperMotor), 468  
 connect() (msl.equipment.resources.picotech.picoscope.enums.PS2000Error), 588  
 method), 588  
`close()` (msl.equipment.resources.thorlabs.kinesis.kconnect\_stateDCServo), 491  
 method), 491  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoscopePS4000A), 491  
`close()` (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoideDCServo), 515  
 method), 515  
 CONNECT\_STATE\_FLOATING  
`close()` (msl.equipment.resources.thorlabs.kinesis.kcube\_steppermotor2KCubeStepperMotor), 524  
 method), 524  
 attribute), 279  
`close()` (msl.equipment.vxi11.RPCClient connect\_to\_ol756()  
 method), 588  
 (msl.equipment.resources.optronic\_laboratories.ol756ocx756ocx), 588  
`close_enum()` (msl.equipment.resources.princeton\_instruments.instrument.PrincetonInstruments), 361  
 static method), 361  
 Connection (class in msl.equipment.connection), 361  
`close_ports()` (msl.equipment.resources.nkt.nktpdll.NKT 20  
 static method), 163  
 connection (msl.equipment.record\_types.EquipmentRecord), 163  
`close_shutter()` attribute), 74  
 (msl.equipment.resources.bentham.benhw32.Bentham32CubeStepperMotor), 122  
 method), 122  
 (class in msl.equipment.connection\_demo), 122  
`close_unit()` (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope), 313  
 method), 313  
 ConnectionMessageBased (class in msl.equipment.connection\_message\_based), 313  
`closedTime` (msl.equipment.resources.thorlabs.kinesis.structs.KinesisClosedTime), 575  
 attribute), 575  
 22  
`CMLError`, 48  
 ConnectionNIDAQ (class in msl.equipment.connection\_nidaq), 48  
`COMMA_SEPERATED` msl.equipment.connection\_nidaq), 48  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000AMetaFormat), 276  
 attribute), 276  
 ConnectionPrologix (class in msl.equipment.connection\_prologix), 276  
`comment` (msl.equipment.record\_types.MaintenanceRecord msl.equipment.connection\_prologix), 276

[27](#)  
**ConnectionPyVISA** (class in [msl.equipment.utils](#)), [583](#)  
[msl.equipment.connection\\_pyvisa](#), [31](#)  
**ConnectionRecord** (class in [msl.equipment.connection](#)), [78](#)  
**connections()** ([msl.equipment.database.Database](#) static method), [20](#)  
**ConnectionSDK** (class in [msl.equipment.utils](#)), [583](#)  
[msl.equipment.connection\\_sdk](#), [32](#)  
**ConnectionSerial** (class in [msl.equipment.utils](#)), [583](#)  
[msl.equipment.connection\\_serial](#), [33](#)  
**ConnectionSocket** (class in [msl.equipment.vxi11](#)), [590](#)  
[msl.equipment.connection\\_socket](#), [35](#)  
**ConnectionTCPIPHisLIP** (class in [msl.equipment.resources.picotech.picoscope.channel.PicoscopeChannel](#)), [222](#)  
[msl.equipment.connection\\_tcpip\\_hislip](#), [36](#)  
**ConnectionTCPIPvXI11** (class in [msl.equipment.vxi11](#)), [590](#)  
[msl.equipment.connection\\_tcpip\\_vxi11](#), [39](#)  
**ConnectionZeroMQ** (class in [msl.equipment.vxi11](#)), [590](#)  
[msl.equipment.connection\\_zeromq](#), [42](#)  
**constants** ([msl.equipment.connection\\_nidaq.ConnectionNIDAQ](#) property), [26](#)  
**constants()** ([msl.equipment.resources.utils.CHeader](#) method), [82](#)  
**context** ([msl.equipment.connection\\_zeromq.ConnectionZeroMQ](#) property), [42](#)  
**continuousCurrentLimit** ([msl.equipment.resources.thorlabs.kinesis.structs.MotionControlSettings](#) attribute), [565](#)  
**controller** ([msl.equipment.connection\\_prologix.ConnectionPrologix](#) property), [28](#)  
**controllers** ([msl.equipment.connection\\_prologix.ConnectionPrologix](#) attribute), [27](#)  
**controlMode** ([msl.equipment.resources.thorlabs.kinesis.structs.MotionControlSettings](#) attribute), [563](#)  
**ControlSettingsType** (class in [msl.equipment.resources.avantes.avaspec](#)), [94](#)  
**controlSrc** ([msl.equipment.resources.thorlabs.kinesis.structs.MotionControlSettings](#) attribute), [565](#)  
**convert\_datetime()** (in module [msl.equipment.resources.omega.ithx](#)), [190](#)  
**convert\_message()** ([msl.equipment.resources.thorlabs.kinesis.motion\\_control.MotionControl](#) static method), [554](#)  
**convert\_to\_date()** (in module [msl.equipment.utils](#)), [583](#)  
**convert\_to\_enum()** (in module [msl.equipment.utils](#)), [582](#)  
**convert\_to\_enum()** ([msl.equipment.connection.Connection](#) static method), [20](#)  
**convert\_to\_primitive()** (in module [msl.equipment.utils](#)), [583](#)  
**convert\_to\_xml\_string()** (in module [msl.equipment.utils](#)), [583](#)  
**copy()** ([msl.equipment.record\\_types.RecordDict](#) method), [80](#)  
**CoreClient** (class in [msl.equipment.vxi11](#)), [590](#)  
**countsPerUnit** ([msl.equipment.resources.thorlabs.kinesis.structs.MotionControlSettings](#) attribute), [557](#)  
**coupling** ([msl.equipment.resources.picotech.picoscope.channel.PicoscopeChannel](#) property), [222](#)  
**CR** ([msl.equipment.connection\\_message\\_based.ConnectionMessageBased](#) attribute), [22](#)  
**create\_callbacks\_file()** (in module [msl.equipment.resources.picotech.picoscope.helper](#)), [311](#)  
**create\_intr\_chan()** ([msl.equipment.connection\\_tcpip\\_vxi11.ConnectionTCPIPvXI11](#) method), [41](#)  
**create\_intr\_chan()** ([msl.equipment.vxi11.CoreClient](#) method), [593](#)  
**create\_zeromq()** ([msl.equipment.vxi11.CoreClient](#) method), [590](#)  
**create\_picoscope\_enums\_file()** (in module [msl.equipment.resources.picotech.picoscope.helper](#)), [311](#)  
**create\_picoscope\_functions\_file()** (in module [msl.equipment.resources.picotech.picoscope.helper](#)), [311](#)  
**create\_tcpip\_nidaq\_structs\_file()** (in module [msl.equipment.resources.picotech.picoscope.helper](#)), [311](#)  
**CtrlFreq** ([msl.equipment.connection\\_nidaq.ConnectionNIDAQ](#) property), [26](#)  
**CtrlTick** ([msl.equipment.connection\\_nidaq.ConnectionNIDAQ](#) property), [26](#)  
**CtrlTime** ([msl.equipment.connection\\_nidaq.ConnectionNIDAQ](#) property), [26](#)  
**ctypes\_map()** (in module [msl.equipment.resources.thorlabs.kinesis.motion\\_control.MotionControl](#)), [311](#)





attribute), 78  
 DateTimeType (class in method), 84  
 msl.equipment.resources.nkt.nktpdll, decrement\_voltage()  
 153 (msl.equipment.resources.aim\_tti.mx\_series.MXSeries  
 Day (msl.equipment.resources.nkt.nktpdll.DateTimeType method), 84  
 attribute), 153 default() (msl.equipment.resources.mks\_instruments.pr4000b.P  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS2000ARatioMode), 12  
 attribute), 235 delay() (msl.equipment.resources.energetiq.eq99.EQ99  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS3000ARatioMode), 136  
 attribute), 251 delete\_group()  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode), 123  
 attribute), 270 (msl.equipment.resources.bentham.benhw32.Bentham32  
 DC (msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode), 136  
 attribute), 293 (msl.equipment.resources.thorlabs.kinesis.structs.Q  
 DC\_1M (msl.equipment.resources.picotech.picoscope.enums.PS1000ARatioMode), 18  
 attribute), 304 (msl.equipment.config.Config attribute), 18  
 DC\_50R (msl.equipment.resources.picotech.picoscope.enums.PS6000ARatioMode), 154  
 attribute), 304 (msl.equipment.resources.nkt.nktpdll.ParameterSetT  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType), 237  
 attribute), 237 (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_B  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS2000VWaveType), 228  
 attribute), 228 DerivFilterOff  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType), 254  
 attribute), 254 (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_De  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType), 275  
 attribute), 275 (msl.equipment.resources.thorlabs.kinesis.enums.P  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS4000WaveType), 265  
 attribute), 265 description PS4000WaveType record\_types.EquipmentRecord  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType), 297  
 attribute), 297 description PS5000WaveType resources.thorlabs.kinesis.structs.TL  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS5000VWaveType), 288  
 attribute), 288 destroy() (msl.equipment.enums.PS5000VWaveType  
 DC\_VOLTAGE (msl.equipment.resources.picotech.picoscope.enums.PS6000VWaveType), 306  
 attribute), 306 destroy\_intr\_chan()  
 deactivate() (msl.equipment.resources.avantes.avaspec.AvaInts equipment.vxi11.CoreClient  
 method), 110 method), 593  
 deadErrorBand (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessCurrentLoopParameters  
 attribute), 559 (msl.equipment.connection\_tcpip\_vxi11.ConnectionTCP  
 DECIMATE (msl.equipment.resources.picotech.picoscope.enums.PS2000ARatioMode), 241  
 attribute), 241 destroy\_link()  
 DECIMATE (msl.equipment.resources.picotech.picoscope.enums.PS3000ARatioMode), 258  
 attribute), 258 (msl.equipment.enums.PS3000ARatioMode), 593  
 DECIMATE (msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode), 280  
 attribute), 280 (msl.equipment.resources.princeton\_instruments.arc\_instr  
 DECIMATE (msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode), 300  
 attribute), 300 det\_read() (msl.equipment.resources.princeton\_instruments.arc\_  
 DECIMATE (msl.equipment.resources.picotech.picoscope.enums.PS5000RatioMode), 291  
 attribute), 291 det\_readall() (msl.equipment.resources.princeton\_instruments.  
 DECIMATE (msl.equipment.resources.picotech.picoscope.enums.PS6000RatioMode), 310  
 attribute), 310 det\_start\_nonblock\_read()  
 decrement\_current() (msl.equipment.resources.princeton\_instruments.arc\_instr

*method*), 362

DetectorType (class in *device\_get\_module\_serial\_number\_str()*  
*msl.equipment.resources.avantes.avaspec*),  
94 (*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 166

device\_abort()  
(*msl.equipment.vxi11.AsyncClient*  
*method*), 593

DEVICE\_CAPABILITY  
(*msl.equipment.resources.picotech.picoscope.enums.PS4000A.MetaType*  
attribute), 275

device\_clear()  
(*msl.equipment.vxi11.CoreClient*  
*method*), 591

device\_clear\_complete()  
(*msl.equipment.hislip.SyncClient*  
*method*), 68

device\_create()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 164

device\_docmd()  
(*msl.equipment.vxi11.CoreClient*  
*method*), 593

device\_enable\_srq()  
(*msl.equipment.vxi11.CoreClient*  
*method*), 592

device\_exists()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 164

device\_get\_all\_types()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
static *method*), 163

device\_get\_boot\_loader\_version()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 164

device\_get\_boot\_loader\_version\_str()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 165

device\_get\_error\_code()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 165

device\_get\_firmware\_version()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 165

device\_get\_firmware\_version\_str()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 165

device\_get\_live()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 166

device\_get\_mode()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 166

*method*), 166

device\_get\_module\_serial\_number\_str()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 166

device\_get\_part\_number\_str()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 166

device\_get\_pcb\_serial\_number\_str()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 167

device\_get\_pcb\_version()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 167

device\_get\_status\_bits()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 167

device\_get\_type()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 167

device\_local()  
(*msl.equipment.vxi11.CoreClient*  
*method*), 592

device\_lock() (*msl.equipment.vxi11.CoreClient*  
*method*), 592

device\_manager() (in module  
*msl.equipment.resources.thorlabs.kinesis.motion\_control*)  
551

device\_meta\_data()  
(*msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope*  
*method*), 339

device\_read() (*msl.equipment.vxi11.CoreClient*  
*method*), 591

device\_readstb()  
(*msl.equipment.vxi11.CoreClient*  
*method*), 591

device\_remote()  
(*msl.equipment.vxi11.CoreClient*  
*method*), 592

device\_remove()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 168

device\_remove\_all()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 168

device\_set\_live()  
(*msl.equipment.resources.nkt.nktpdll.NKT*  
*method*), 168

DEVICE\_SETTINGS  
(*msl.equipment.resources.picotech.picoscope.enums.PS4000A.MetaType*  
attribute), 275

device\_trigger()

|   |   |
|---|---|
| <i>(msl.equipment.vxi11.CoreClient</i><br><i>method)</i> , 591  | DeviceModeChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160         |
| device_unlock()<br><i>(msl.equipment.vxi11.CoreClient</i><br><i>method)</i> , 592   | DeviceModeTypes (class in<br><i>msl.equipment.resources.nkt.nktpdll)</i> ,<br>154   |
| device_write()<br><i>(msl.equipment.vxi11.CoreClient</i><br><i>method)</i> , 590  | DeviceModuleSerialChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155     |
| DeviceBlVerChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155                      | DeviceModuleSerialChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160 |
| DeviceBlVerChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160                  | DevicePartNumberChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155       |
| DeviceClearAcknowledge (class in<br><i>msl.equipment.hislip)</i> , 59   | DevicePartNumberChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160   |
| DeviceClearAcknowledge<br><i>(msl.equipment.hislip.MessageType</i><br><i>attribute)</i> , 51  | DevicePCBSerialChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155        |
| DeviceClearComplete (class in<br><i>msl.equipment.hislip)</i> , 59  | DevicePCBSerialChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160    |
| DeviceClearComplete<br><i>(msl.equipment.hislip.MessageType</i><br><i>attribute)</i> , 51   | DevicePCBVersionChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155       |
| DeviceConfigType (class in<br><i>msl.equipment.resources.avantes.avaspec)</i> ,<br>99   | DevicePCBVersionChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160   |
| deviceDependantData<br><i>(msl.equipment.resources.thorlabs.kinesis.structs.TLH_HardwareInformation)</i><br><i>attribute)</i> , 555 | DeviceStatus (class in<br><i>msl.equipment.resources.avantes.avaspec)</i> ,<br>91   |
| DeviceErrorCodeChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155                  | DeviceStatusBitsChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155       |
| DeviceErrorCodeChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160              | DeviceStatusBitsChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160   |
| DeviceFwVerChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155                      | DeviceStatusCallback (in module<br><i>msl.equipment.resources.nkt.nktpdll)</i> ,<br>153                                   |
| DeviceFwVerChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160                  | DeviceStatusCallback<br><i>(msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>attribute)</i> , 159                        |
| DeviceLiveChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155                       | DeviceStatusTypes (class in<br><i>msl.equipment.resources.nkt.nktpdll)</i> ,<br>154                                       |
| DeviceLiveChanged<br><i>(msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusTypes</i><br><i>attribute)</i> , 160                   | DeviceSysTypeChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155          |
| DeviceModeChanged<br><i>(msl.equipment.resources.nkt.nktpdll.DeviceStatusTypes</i><br><i>attribute)</i> , 155                       |   |

|                      |   |
|----------------------|---|
| DeviceSysTypeChanged | dewpoint() (msl.equipment.resources.omega.ithx.iTHX (msl.equipment.resources.nkt.nktpdll.NKT.DeviceStatusType), 187 |
| DeviceTypeChanged    | diameter (msl.equipment.resources.thorlabs.kinesis.structs.NT_C attribute), 160                                     |
| DeviceTypeChanged    | DIFFERENTIAL_TO_115MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 155                           |
| DeviceTypeChanged    | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 160                      |
| DevModeAnalyze       | DIFFERENTIAL_TO_115MV (msl.equipment.resources.picotech.pt104.Pt104DataType attribute), 358                         |
| DevModeAnalyze       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.picotech.pt104.Pt104DataType attribute), 358                        |
| DevModeAnalyze       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 154                          |
| DevModeAnalyze       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.MOT_B attribute), 559                      |
| DevModeAnalyzeInit   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeAnalyzeInit   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.MOT_D attribute), 561                      |
| DevModeAnalyzeInit   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 154                          |
| DevModeAnalyzeInit   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.MOT_P attribute), 566                      |
| DevModeAnalyzeInit   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeAnalyzeInit   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.QD_Loc attribute), 573                     |
| DevModeDisabled      | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 154                          |
| DevModeDisabled      | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.QD_PIL attribute), 573                     |
| DevModeDisabled      | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeDisabled      | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.TC_Loo attribute), 578                     |
| DevModeError         | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 154                          |
| DevModeError         | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeError         | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeLogDownload   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeLogDownload   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeLogDownload   | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeNormal        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 154                      |
| DevModeNormal        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeNormal        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeNormal        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeTimeout       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 154                          |
| DevModeTimeout       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeTimeout       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeTimeout       | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.thorlabs.kinesis.structs.FF_IOS attribute), 566                     |
| DevModeUpload        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.DeviceModeType attribute), 154                          |
| DevModeUpload        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.picotech.picoscope.structs.PS2 attribute), 346                      |
| DevModeUpload        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeUpload        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.picotech.picoscope.structs.PS2 attribute), 346                      |
| DevModeUpload        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.nkt.nktpdll.NKT.DeviceModeType attribute), 159                      |
| DevModeUpload        | DIFFERENTIAL_TO_2500MV (msl.equipment.resources.picotech.picoscope.structs.PS3 attribute), 346                      |





attribute), 569  
 displayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KLBeamMIPParams attribute), 569  
 DisplayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KMOTMMIPParams attribute), 567  
 DisplayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KNA4MMIPParams attribute), 571  
 DisplayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KPZ4MMIPParams attribute), 572  
 DisplayIntensity (msl.equipment.resources.thorlabs.kinesis.structs.KSG4MMIPParams attribute), 577  
 displayMode (msl.equipment.resources.thorlabs.kinesis.structs.KTSCIOSettings attribute), 576  
 displays\_enable() (msl.equipment.resources.mks\_instruments.DONT\_CARE4000B attribute), 142  
 displays\_setup() (msl.equipment.resources.mks\_instruments.pr4000b.PR4000B attribute), 143  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KMOTMMIPParams attribute), 567  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KPZ4MMIPParams attribute), 572  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KSG4MMIPParams attribute), 576  
 DisplayTimeout (msl.equipment.resources.thorlabs.kinesis.structs.KSG4MMIPParams attribute), 577  
 DISTANCE (msl.equipment.resources.thorlabs.kinesis.enums.UnitType attribute), 462  
 DISTRIBUTION (msl.equipment.resources.picotech.picoscope.enums.PS4000ARatioMode attribute), 280  
 DISTRIBUTION (msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode attribute), 301  
 DISTRIBUTION (msl.equipment.resources.picotech.picoscope.enums.PS5000ARatioMode attribute), 291  
 DISTRIBUTION (msl.equipment.resources.picotech.picoscope.enums.PS6000ARatioMode attribute), 310  
 dmm\_factory() (in module msl.equipment.resources.dmm), 81  
 do\_averaging() (msl.equipment.resources.optronic\_laboratories.ol756ocx.Ol756ocx attribute), 477  
 do\_calculations() (msl.equipment.resources.optronic\_laboratories.ol756ocx.Ol756ocx attribute), 41  
 docmd() (msl.equipment.connection\_tcpip\_vxll.ConnectionTCPIP method), 41  
 done() (msl.equipment.resources.avantes.avaspec.Avantes attribute), 113  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 241  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 240  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 241  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 230  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 279  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 279  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 300  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 291  
 DONT\_CARE (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 291  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 236  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS2000A attribute), 228  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 253  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 274  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 274  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 274  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 274  
 DOWN (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 274  
 downLimit (msl.equipment.resources.thorlabs.kinesis.structs.KNA4MMIPParams attribute), 570  
 downLimit (msl.equipment.resources.thorlabs.kinesis.structs.NT\_T

attribute), 562

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS2000ASweepType attribute), 236

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS2000ASweepType attribute), 228

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS3000ASweepType attribute), 253

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS4000ASweepType attribute), 274

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS4000ASweepType attribute), 264

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS5000ASweepType attribute), 296

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS5000ASweepType attribute), 287

DOWNUP (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 305

DRIVER\_VERSION (msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfoApi attribute), 224

DRIVER\_VERSION (msl.equipment.resources.picotech.picoscope.enums.PS2000Info attribute), 226

DRIVER\_VERSION (msl.equipment.resources.picotech.picoscope.enums.PS3000Info attribute), 244

DRV013\_25MM (msl.equipment.resources.thorlabs.kinesis.enums.KST\_Steps attribute), 457

DRV014\_50MM (msl.equipment.resources.thorlabs.kinesis.enums.KST\_Steps attribute), 457

dt (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope property), 312

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndexMode attribute), 239

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS3000AIndexMode attribute), 255

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndexMode attribute), 277

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndexMode attribute), 266

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndexMode attribute), 298

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndexMode attribute), 289

DUAL (msl.equipment.resources.picotech.picoscope.enums.PS6000AIndexMode attribute), 308

DynamicStorageType (class in msl.equipment.resources.avantes.avaspec), 97

E

E\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000ASweepType attribute), 271

E\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS2000ASweepType attribute), 271

E\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS3000ASweepType attribute), 271

EDGE\_TRIGGER\_SOURCE (msl.equipment.resources.avantes.avaspec.Avantes attribute), 401

EF\_10MHZ (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 302

EF\_20MHZ (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 302

EF\_25MHZ (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 302

EF\_5MHZ (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 302

EF\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 302

EF\_OFF (msl.equipment.resources.picotech.picoscope.enums.PS6000ASweepType attribute), 302

EIGHT (msl.equipment.constants.DataBits attribute), 45

enable\_calibration\_file() (msl.equipment.resources.optronic\_laboratories.ol756ocx method), 438

enable\_channel() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 390

enable\_channel() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper method), 469

enable\_channel() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo method), 491

enable\_channel() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper method), 394

enable\_dark\_current() (msl.equipment.resources.optronic\_laboratories.ol756ocx method), 198

enable\_last\_msg\_timer() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 394

enable\_last\_msg\_timer() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper method), 466

enable\_last\_msg\_timer() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper method), 469

624 Index



method), 579

ETH\_ALREADY\_IN\_USE\_USB (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

errcheck\_negative() (msl.equipment.resources.thorlabs.fwx2c.FilterWheelAttribute), 580

ETH\_ALREADY\_IN\_USE\_USB (msl.equipment.resources.avantes.avaspec.DeviceStatus attribute), 102

errcheck\_negative\_one() (msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k attribute), 316

ETH\_AVAILABLE (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

errcheck\_non\_zero() (msl.equipment.resources.thorlabs.fwx2c.FilterWheelAttribute), 580

ETH\_AVAILABLE (msl.equipment.resources.avantes.avaspec.DeviceStatus attribute), 92

errcheck\_one() (msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k attribute), 316

ETH\_CONN\_STATUS\_CONNECTED (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

errcheck\_true() (msl.equipment.resources.thorlabs.kinesis.motion\_controller.MotionController attribute), 553

ETH\_CONN\_STATUS\_CONNECTED\_NOMON (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

errcheck\_zero() (msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k attribute), 316

ETH\_CONN\_STATUS\_CONNECTING (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

Error, 52

ETH\_CONN\_STATUS\_NOCONNECTION (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

error (msl.equipment.hislip.AsyncEndTLSResponse property), 65

ETH\_IN\_USE\_BY\_APPLICATION (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

error (msl.equipment.hislip.AsyncLockResponse property), 56

ETH\_IN\_USE\_BY\_APPLICATION (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

error (msl.equipment.hislip.AsyncStartTLSResponse property), 64

ETH\_IN\_USE\_BY\_APPLICATION (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 92

error (msl.equipment.hislip.AuthenticationResult property), 66

ETH\_IN\_USE\_BY\_OTHER (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

Error (msl.equipment.hislip.MessageType attribute), 50

ETH\_IN\_USE\_BY\_OTHER (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

error\_code (msl.equipment.hislip.AuthenticationResult property), 67

ETH\_IN\_USE\_BY\_OTHER (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 92

ERROR\_CODE (msl.equipment.resources.picotech.picoscope.enums.PS2000Info attribute), 226

ETHERNETSERIALPORTTYPE (class in msl.equipment.resources.avantes.avaspec), 400

ERROR\_CODES (msl.equipment.resources.mks\_instruments.pr4000b.PR4000B attribute), 142

ETS (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 284

error\_to\_english() (msl.equipment.resources.princeton\_instruments.picoscope.picoscope2k3k.PicoScope2k3k static method), 387

ETS\_CYCLE (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 284

ErrorHandler (msl.equipment.resources.nkt.nkt\_dll.nkt\_dll attribute), 154

ETS\_INTERFACE (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 285

ErrorMessage (class in msl.equipment.hislip), 54

ETS\_SAMPLE\_TIME\_PICOSECONDS (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 285

errors (msl.equipment.connection\_nidaq.ConnectionNIDAQ property), 26

ETS\_STATE (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 284

ErrorType (class in msl.equipment.hislip), 52

ETH7010 (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 102

ETH7010 (msl.equipment.resources.avantes.avaspec.Avantes.DeviceStatus attribute), 92

EVEN (msl.equipment.constants.Parity attribute), 44

EVENT (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes.picotech.picoscope.enums.PS5000CHoldTypes attribute), 259

event\_register() (msl.equipment.resources.picotech.picoscope.enums.PS6000CHoldTypes attribute), 285

event\_status\_register() (msl.equipment.resources.energetiq.eq99.EQ99 attribute), 302

excessEnergyLimit (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 238

exclusive (msl.equipment.resources.aim\_tti.mx\_series.EMXSeries attribute), 255

EXIT (msl.equipment.resources.thorlabs.kinesis.structs.MOTFrunchLesElectricOutputParameters attribute), 255

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 265

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 265

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 240

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 229

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes attribute), 256

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes attribute), 246

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 278

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 267

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 299

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 290

EXIT (msl.equipment.resources.picotech.picoscope.enums.PS6000AHoldTypes attribute), 309

export\_config\_file() (msl.equipment.resources.picotech.picoscope.ps5000a.PicotechPS5000a attribute), 199

export\_registry() (msl.equipment.resources.optronic\_laboratories.ol756.ol756 attribute), 199

EXT (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 231

EXT (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 225

EXT (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes attribute), 249

EXT (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes attribute), 242

EXT (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 271

EXT (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 260

EXT (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 294

EXT (msl.equipment.resources.thorlabs.kinesis.structs.MOTFrunchLesElectricOutputParameters attribute), 255

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 265

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 265

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 240

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS2000AHoldTypes attribute), 229

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes attribute), 256

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldTypes attribute), 246

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 278

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS4000AHoldTypes attribute), 267

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 299

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS5000AHoldTypes attribute), 290

EXT\_IN (msl.equipment.resources.picotech.picoscope.enums.PS6000AHoldTypes attribute), 309

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps2000a.PicotechPS2000a attribute), 331

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps3000a.PicotechPS3000a attribute), 334

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicotechPS4000a attribute), 337

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicotechPS4000a attribute), 339

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps5000a.PicotechPS5000a attribute), 340

EXT\_MAX\_VALUE (msl.equipment.resources.picotech.picoscope.ps5000a.PicotechPS5000a attribute), 341

EXT\_MAX\_VALUE (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IOChannel attribute), 434

EXT\_MAX\_VOLTAGE (msl.equipment.resources.picotech.picoscope.ps2000a.PicotechPS2000a attribute), 346

EXT\_MAX\_VOLTAGE (msl.equipment.resources.picotech.picoscope.ps3000a.PicotechPS3000a attribute), 346

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS2000PwrCond) (msl.equipment.resources.nkt.nktpdll.ParameterSetType attribute), 346

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS2000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS2000PwrCond attribute), 345

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS3000APresCond) (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 349

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS3000APresCond) (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 350

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS3000APresCond) (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 349

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS3000APresCond) (msl.equipment.resources.picotech.picoscope.enums.PS3000A attribute), 349

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS3000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS3000PwrCond attribute), 348

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS3000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS3000PwrCond attribute), 348

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS4000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS4000PwrCond attribute), 352

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS4000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS4000PwrCond attribute), 351

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS5000APresCond) (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 356

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS5000APresCond) (msl.equipment.resources.picotech.picoscope.enums.PS5000A attribute), 355

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS5000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS5000PwrCond attribute), 354

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS5000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS5000PwrCond attribute), 354

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS6000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS6000PwrCond attribute), 357

external (msl.equipment.resources.picotech.picoscope.factory.VA\_PS6000PwrCond) (msl.equipment.resources.picotech.picoscope.enums.PS6000PwrCond attribute), 356

external\_input() (msl.equipment.resources.mks\_instruments.pr4000b.PR4000B attribute), 143

EXTERNAL\_TRIGGER (msl.equipment.resources.avantes.avaspec.AVANTES attribute), 101

ExtSignalSMA (msl.equipment.resources.thorlabs.kinesis.structs.HALFPIGLogMod attribute), 453

F (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 271

F\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 271

F\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 271

factorForOutput (msl.equipment.resources.thorlabs.kinesis.structs.MQTBuff attribute), 559

FALLING (msl.equipment.resources.picotech.picoscope.enums.PS4000A attribute), 290

FALLING (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 307

FALLING (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 308

FALLING (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 240

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 255

FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS6000A attribute), 255

attribute), 278  
 FALLING\_LOWER (msl.equipment.resources.picotech.fib.kinipic.switch.PS4000ThresholdDirection  
 attribute), 267  
 FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection  
 attribute), 299  
 FALLING\_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection  
 attribute), 309  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS2000AIE), 269  
 attribute), 241  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS2000AIE), 285  
 attribute), 230  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS2000AIE), 269  
 attribute), 257  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS2000AIE), 275  
 attribute), 247  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS4000AIE), 269  
 attribute), 279  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS4000AIE), 275  
 attribute), 267  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS5000AIE), 269  
 attribute), 300  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS5000AIE), 285  
 attribute), 291  
 FALSE (msl.equipment.resources.picotech.picoscope.enums.PS6000AIE), 265  
 attribute), 309  
 FAST (msl.equipment.resources.picotech.picoscope.enums.PS2000AIE), 235  
 attribute), 227  
 FAST (msl.equipment.resources.picotech.picoscope.enums.PS3000AIE), 252  
 attribute), 245  
 FAST (msl.equipment.resources.picotech.picoscope.enums.PS4000AIE), 273  
 attribute), 263  
 FAST (msl.equipment.resources.picotech.picoscope.enums.PS5000AIE), 295  
 attribute), 287  
 FAST (msl.equipment.resources.picotech.picoscope.enums.PS6000AIE), 304  
 attribute), 579  
 FatalError, 52  
 FatalError (msl.equipment.hislip.MessageType  
 attribute), 50  
 FatalErrorMessage (class in  
 msl.equipment.hislip), 53  
 FB\_Encoder (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_FB\_Encoder), 440  
 attribute), 440  
 FB\_EncoderSwitch (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_FB\_EncoderSwitch), 440  
 attribute), 440  
 FC\_20 (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection), 269  
 FC\_200 (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection), 269  
 FC\_20K (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection), 269  
 FC\_2K (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection), 269  
 FC\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ThresholdDirection), 269  
 feature\_bitmap (msl.equipment.resources.thorlabs.kinesis.structs.BNT\_IO), 59  
 feedPS2000 (msl.equipment.resources.thorlabs.kinesis.structs.PF), 76  
 feedbackSrc (msl.equipment.resources.thorlabs.kinesis.structs.PF), 76  
 feedForward (msl.equipment.resources.thorlabs.kinesis.structs.M), 76  
 FF\_InputButton (msl.equipment.resources.thorlabs.kinesis.enums.FF\_Sign), 437  
 FF\_InputButton (msl.equipment.resources.thorlabs.kinesis.enums.FF\_Sign), 437  
 FF\_InputButton (msl.equipment.resources.thorlabs.kinesis.enums.FF\_Sign), 437  
 FF\_IOSettings (class in  
 msl.equipment.resources.thorlabs.kinesis.enums), 436  
 FF\_IOSettings (class in  
 msl.equipment.resources.thorlabs.kinesis.structs), 566  
 FF\_OutputHighAtSetPosition (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOSettings), 436  
 FF\_OutputHighAtSetPosition (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOSettings), 436  
 FF\_OutputHighAtSetPosition (msl.equipment.resources.thorlabs.kinesis.enums.FF\_IOSettings), 436



(`msl.equipment.resources.thorlabs.kinesis.enums.FF_SignalModes`), 19  
 attribute), 436 `find()` (`msl.equipment.resources.avantes.avaspec.Avantes`  
 static method), 113  
`FF_OutputLevel` (`msl.equipment.resources.thorlabs.kinesis.enums.FF_OutputLevel`) (in module  
 attribute), 437 `msl.equipment.factory`), 50  
`FF_OutputPulse` `find_lxi()` (in module  
 (`msl.equipment.resources.thorlabs.kinesis.enums.FF_OutputPulse`), 48  
 attribute), 437 `find_signal_modes()` (`msl.equipment.factory`), 50  
`FF_OutputSwap` (`msl.equipment.resources.thorlabs.kinesis.enums.FF_OutputSwap`) (in module  
 attribute), 437 `msl.equipment.connection_prologix`), 30  
`FF_PositionError` (`msl.equipment.resources.thorlabs.kinesis.enums.FF_PositionError`) (in module  
 attribute), 436 `msl.equipment.resources`), 81  
`FF_Positions` (class in `find_vxi11()` (in module `msl.equipment.vxi11`),  
`msl.equipment.resources.thorlabs.kinesis.enums`), 594  
 436 `findall()` (`msl.equipment.config.Config`  
 method), 19  
`FF_SetPositionOnPositiveEdge` (`msl.equipment.resources.thorlabs.kinesis.enums.FF_SetPositionOnPositiveEdge`) (in module  
 attribute), 436 `msl.equipment.resources.picotech.picoscope.enums.PicoScopeEnums`), 224  
`FF_SignalModes` (class in `msl.equipment.resources.thorlabs.kinesis.enums.FF_SignalModes`) (in module  
 attribute), 436 `msl.equipment.resources.picotech.picoscope.enums.PicoScopeEnums`), 224  
`FF_ToggleOnPositiveEdge` (`msl.equipment.resources.thorlabs.kinesis.enums.FF_ToggleOnPositiveEdge`) (in module  
 attribute), 436 `msl.equipment.resources.thorlabs.kinesis.structs.TLI_HardwareVersion`), 224  
`filter1Fc` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`) (in module  
 attribute), 565 `FIRST_USB` (`msl.equipment.resources.picotech.picoscope.ps2000.PicoScopeEnums`), 224  
`filter1Q` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`) (in module  
 attribute), 565 `FIRST_USB` (`msl.equipment.resources.picotech.picoscope.ps3000.PicoScopeEnums`), 224  
`filter2Fc` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`) (in module  
 attribute), 565 `FIVE` (`msl.equipment.constants.DataBits` at  
 attribute), 565 `flash_led()` (`msl.equipment.resources.picotech.picoscope.picoscope.enums.PicoScopeEnums`), 224  
`Filter_Flipper` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`) (in module  
 attribute), 551 `Flash_led()` (`msl.equipment.resources.picotech.picoscope.picoscope.enums.PicoScopeEnums`), 224  
`filter_home()` (`msl.equipment.resources.princeton.force_calibration.instrument.PrincetonInstruments`  
 method), 362 (`msl.equipment.resources.thorlabs.kinesis.structs.TSG_IO`), 224  
`Filter_Wheel` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`) (in module  
 attribute), 551 `Forward` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_Trigger`), 224  
`FilterCount` (class in `msl.equipment.resources.thorlabs.fwx2c`), `Forward` (`msl.equipment.resources.thorlabs.kinesis.enums.TIM_Direction`), 224  
 578 attribute), 460  
`FilterFlipper` (class in `Forwards` (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_Direction`), 224  
`msl.equipment.resources.thorlabs.kinesis.filter_flipper`), 424  
 463 `FPBrightness` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`), 224  
`filterNo` (`msl.equipment.resources.thorlabs.kinesis.structs.PicoScopeEnums`) (in module  
 attribute), 565 `FREQUENCY_COUNTER`  
`FilterWheelXX2C` (class in `msl.equipment.resources.picotech.picoscope.enums.PS4000Enums`), 283  
 579 `FREQUENCY_COUNTER_CHANNEL`

(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 283 FT\_InvalidParameter  
 FREQUENCY\_COUNTER\_ENABLED (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 283 FT\_IOError (msl.equipment.resources.thorlabs.kinesis.enums.FT\_  
 FREQUENCY\_COUNTER\_RANGE (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 283 attribute), 419  
 FREQUENCY\_COUNTER\_THRESHOLDMAJOR FT\_Status (class in  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 283 418  
 FREQUENCY\_COUNTER\_THRESHOLDMINOR functions() (msl.equipment.resources.utils.CHeader  
 (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue  
 attribute), 283 futureUse (msl.equipment.resources.thorlabs.kinesis.structs.TSG\_  
 from\_bytes() (in module msl.equipment.utils), attribute), 576  
 585 FW103 (msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stage  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS2000Attribute), 236 FW103 (msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stage  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS2000Attribute), 226 FW\_FAIL (msl.equipment.resources.picotech.errors.PS2000Error  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS3000Attribute), 252 FW\_FAIL (msl.equipment.resources.picotech.errors.PS3000Error  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS3000Attribute), 244 attribute), 320  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS4000ATimeUnits  
 attribute), 273 G  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS4000Attribute), 263 G (msl.equipment.resources.picotech.picoscope.enums.PS4000ACh  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS5000Attribute), 296 G\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS5000Attribute), 287 G\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000  
 FS (msl.equipment.resources.picotech.picoscope.enums.PS6000TimeUnits  
 attribute), 305 GAIN (msl.equipment.resources.cmi.sia3.SIA3 at-  
 FT\_DeviceNotFound gain (msl.equipment.resources.thorlabs.kinesis.structs.NT\_GainPa  
 (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status), 419 attribute), 563  
 FT\_DeviceNotOpened GARBAGE\_ARGS (msl.equipment.vxi11.AcceptStatus  
 (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status), 419 attribute), 587  
 FT\_DeviceNotPresent GATE\_HIGH (msl.equipment.resources.picotech.picoscope.enums.PS  
 (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status), 419 attribute), 238  
 FT\_IncorrectDevice GATE\_HIGH (msl.equipment.resources.picotech.picoscope.enums.PS  
 (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status), 419 attribute), 254  
 FT\_InsufficientResources GATE\_HIGH (msl.equipment.resources.picotech.picoscope.enums.PS  
 (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status), 419 attribute), 276  
 FT\_InvalidHandle GATE\_HIGH (msl.equipment.resources.picotech.picoscope.enums.PS  
 (msl.equipment.resources.thorlabs.kinesis.enums.FT\_Status), 307 attribute), 288

GATE\_LOW (msl.equipment.resources.picotech.picoscope.enums.PS2000ASigGenTrigType attribute), 238

GD\_Trig\_Low (msl.equipment.resources.thorlabs.kinesis.enums.QDTrigType attribute), 254

GATE\_LOW (msl.equipment.resources.picotech.picoscope.enums.PS4000ASigGenTrigType attribute), 276

GATE\_LOW (msl.equipment.resources.picotech.picoscope.enums.PS4000ASigGenTrigType attribute), 265

GATE\_LOW (msl.equipment.resources.picotech.picoscope.enums.PS5000ASigGenTrigType attribute), 297

GATE\_LOW (msl.equipment.resources.picotech.picoscope.enums.PS5000ASigGenTrigType attribute), 289

GATE\_LOW (msl.equipment.resources.picotech.picoscope.enums.PS6000ASigGenTrigType attribute), 307

gateway (msl.equipment.connection\_sdk.ConnectionSDK property), 32

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 237

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType attribute), 228

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 253

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 275

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 264

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 297

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 288

GAUSSIAN (msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType attribute), 306

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps2000m.PhasedScope attribute), 332

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000m.PhasedScope attribute), 335

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps4000m.PhasedScope attribute), 339

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000m.PhasedScope attribute), 341

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000m.PhasedScope attribute), 342

GAUSSIAN\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps6000m.PhasedScope attribute), 344

GD\_Trig\_High (msl.equipment.resources.thorlabs.kinesis.enums.QDTrigType attribute), 254

GenericDCMotor (in module msl.equipment.resources.thorlabs.kinesis.messages), 549

GenericMotor (in module msl.equipment.resources.thorlabs.kinesis.messages), 549

GenericSimpleMotor (in module msl.equipment.resources.thorlabs.kinesis.messages), 550

GenericWaveType (in module msl.equipment.resources.thorlabs.kinesis.messages), 550

get (msl.equipment.resources.bentham.benhw32.Bentham32 method), 122

get (msl.equipment.resources.bentham.benhw64.Bentham method), 124

get\_access\_channel() (msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX method), 143

get\_adaptive\_int\_time\_index() (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b method), 131

get\_all\_ports() (msl.equipment.resources.nkt.nktpdll.NKT static method), 168

get\_analog\_in() (msl.equipment.resources.avantes.avaspec.Avantes method), 110

get\_analogue\_offset() (msl.equipment.resources.picotech.picoscope.picoscope\_a method), 319

get\_backlash() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 391

get\_is\_backlash() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 391

get\_KPA\_TriggerPolarities (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 391





`method`), 203  
`get_dark_current_wavelength()`  
 (`msl.equipment.resources.optronic_laboratories.ol7500a.Ol7500a`  
`method`), 203  
`get_dark_pixel_data()`  
 (`msl.equipment.resources.avantes.avaspec.Avantes`  
`method`), 112  
`get_data()` (`msl.equipment.resources.avantes.avaspec.Avantes`  
`method`), 115  
`get_dcpid_params()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo`  
`method`), 491  
`get_dead_band()`  
 (`msl.equipment.resources.mks_instruments.pr4000b.PR4000B`  
`method`), 143  
`get_descriptors()`  
 (`msl.equipment.hislip.HiSLIPClient`  
`method`), 67  
`get_det_bipolar()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 367  
`get_det_bipolar_str()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 367  
`get_det_hv_on()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 367  
`get_det_hv_volts()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 367  
`get_det_num_avg_read()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 368  
`get_det_range()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 368  
`get_det_range_factor()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 368  
`get_det_type()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 368  
`get_det_type_str()`  
 (`msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments`  
`method`), 368  
`get_device_info()`  
 (`msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl`  
`static method`), 554  
`get_device_list()`  
 (`msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl`  
`static method`), 554  
`get_device_list_size()`  
 (`msl.equipment.resources.thorlabs.kinesis.motion_control.MotionControl`  
`static method`), 554  
`get_device_resolution()`  
 (`msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope`  
`method`), 342  
`get_device_unit_from_real_value()`  
 (`msl.equipment.resources.thorlabs.kinesis.benchtop_stepper.BenchtopStepper`  
`method`), 391  
`get_device_unit_from_real_value()`  
 (`msl.equipment.resources.thorlabs.kinesis.integrated_stepper.IntegratedStepper`  
`method`), 470  
`get_device_unit_from_real_value()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo`  
`method`), 491  
`get_device_unit_from_real_value()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_stepper.KCubeStepper`  
`method`), 525  
`get_dialog()` (`msl.equipment.resources.mks_instruments.pr4000b.PR4000B`  
`method`), 144  
`get_digital_in()`  
 (`msl.equipment.resources.avantes.avaspec.Avantes`  
`method`), 111  
`get_digital_outputs()`  
 (`msl.equipment.resources.thorlabs.kinesis.benchtop_stepper.BenchtopStepper`  
`method`), 392  
`get_digital_outputs()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo`  
`method`), 492  
`get_digital_outputs()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KCubeSolenoid`  
`method`), 511  
`get_digital_outputs()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_stepper.KCubeStepper`  
`method`), 526  
`get_display_text()`  
 (`msl.equipment.resources.mks_instruments.pr4000b.PR4000B`  
`method`), 144  
`get_dll_version()`  
 (`msl.equipment.resources.avantes.avaspec.Avantes`  
`method`), 111  
`get_encoder_counter()`  
 (`msl.equipment.resources.thorlabs.kinesis.benchtop_stepper.BenchtopStepper`  
`method`), 392  
`get_encoder_counter()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo`  
`method`), 492  
`get_encoder_counter()`  
 (`msl.equipment.resources.thorlabs.kinesis.kcube_stepper.KCubeStepper`  
`method`), 526  
`get_ending_wavelength()`

|  |  |
|--|--|
| <code>(msl.equipment.resources.optronic_laboratories.ol756ocx.Ol756ocx</code>                      | <code>(msl.equipment.resources.thorlabs.kinesis.integrated_step</code>                             |
| <code>method), 203</code>  | <code>method), 470</code>  |
| <code>get_enum_preopen_com()</code>  | <code>get_formula_relay()</code>   |
| <code>(msl.equipment.resources.princeton_instruments.arc_instrument.pr4000b.PR4000b</code>         | <code>(msl.equipment.princeton_instruments.pr4000b.PR4000b</code>                                  |
| <code>static method), 370</code>   | <code>method), 144</code>  |
| <code>get_enum_preopen_model()</code>  | <code>get_formula_temporary()</code>   |
| <code>(msl.equipment.resources.princeton_instruments.arc_instrument.pr4000b.PR4000b</code>         | <code>(msl.equipment.princeton_instruments.pr4000b.PR4000b</code>                                  |
| <code>static method), 369</code>   | <code>method), 144</code>  |
| <code>get_enum_preopen_serial()</code>   | <code>get_front_panel_locked()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.arc_instrument.pr4000b.PR4000b</code>         | <code>(msl.equipment.thorlabs.kinesis.kcube_dc_server.KcubeDcServer</code>                         |
| <code>static method), 370</code>   | <code>method), 492</code>  |
| <code>get_exposure_mode()</code>   | <code>get_front_panel_locked()</code>  |
| <code>(msl.equipment.resources.energetiq.eq99.EQ99</code>  | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KcubeStepperMotor</code>       |
| <code>method), 138</code>  | <code>method), 526</code>  |
| <code>get_exposure_time()</code>   | <code>get_gain()</code>  |
| <code>(msl.equipment.resources.energetiq.eq99.EQ99</code>  | <code>(msl.equipment.resources.mks_instruments.pr4000b.PR4000b</code>                              |
| <code>method), 137</code>  | <code>method), 144</code>  |
| <code>get_external_input_range()</code>  | <code>get_gain_index()</code>  |
| <code>(msl.equipment.resources.mks_instruments.pr4000b.PR4000b</code>                              | <code>(msl.equipment.resources.optronic_laboratories.ol756ocx.Ol756ocx</code>                      |
| <code>method), 144</code>  | <code>method), 203</code>  |
| <code>get_external_output_range()</code>   | <code>get_group()</code>   |
| <code>(msl.equipment.resources.mks_instruments.pr4000b.PR4000b</code>                              | <code>(msl.equipment.resources.bentham.benhw32.BenthamBenchtopStepperMotor</code>                  |
| <code>method), 144</code>  | <code>method), 122</code>  |
| <code>get_filter_max_pos()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.avantes.avaspec.Avantes</code>                                      |
| <code>method), 369</code>  | <code>method), 113</code>  |
| <code>get_filter_min_pos()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor</code> |
| <code>method), 369</code>  | <code>method), 392</code>  |
| <code>get_filter_model()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper</code>                |
| <code>method), 369</code>  | <code>method), 463</code>  |
| <code>get_filter_position()</code>   | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.thorlabs.kinesis.integrated_step</code>                             |
| <code>method), 369</code>  | <code>method), 470</code>  |
| <code>get_filter_preopen_model()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KcubeDcServer</code>               |
| <code>static method), 369</code>   | <code>method), 492</code>  |
| <code>get_filter_present()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KcubeSolenoid</code>                |
| <code>method), 369</code>  | <code>method), 515</code>  |
| <code>get_filter_serial()</code>   | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.princeton_instruments.princeton_instruments</code>                  | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KcubeStepperMotor</code>       |
| <code>method), 369</code>  | <code>method), 526</code>  |
| <code>get_firmware_version()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor</code> | <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor</code> |
| <code>method), 392</code>  | <code>method), 392</code>  |
| <code>get_firmware_version()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.integrated_step</code>                             | <code>(msl.equipment.resources.thorlabs.kinesis.integrated_step</code>                             |
| <code>method), 470</code>  | <code>method), 470</code>  |
| <code>get_firmware_version()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper</code>                | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KcubeDcServer</code>               |
| <code>method), 464</code>  | <code>method), 492</code>  |
| <code>get_firmware_version()</code>  | <code>get_group_serial()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KcubeDcServer</code>               | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KcubeDcServer</code>               |
| <code>method), 492</code>  | <code>method), 492</code>  |

`get_hardware_info_block()` (*msl.equipment.resources.mks\_instruments.pr4000b.PR4000b.KCuboSolenoid*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCuboSolenoid*  
*method), 516* `get_input_status()`  
`get_hardware_info_block()` (*msl.equipment.resources.optosigma.shot702.SHOT702*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*method), 526* `get_input_voltage()`  
`get_hardware_type()` (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.KCubeStepperMotor*  
*(msl.equipment.resources.bentham.benhw32.Bentham32*  
*method), 393* `get_integration_time_index()`  
`get_hardware_type()` (*msl.equipment.resources.optronic\_laboratories.ol756ocx.ol756ocx*  
*(msl.equipment.resources.bentham.benhw64.Bentham64*  
*method), 124* `get_interface_mode()`  
`get_homing_params_block()` (*msl.equipment.resources.mks\_instruments.pr4000b.PR4000b*  
*(msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.KCubeStepperMotor*  
*method), 392* `get_io_settings()`  
`get_homing_params_block()` (*msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper*  
*(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors*  
*method), 471* `get_ip_config()`  
`get_homing_params_block()` (*msl.equipment.resources.avantes.avaspec.Avantes*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo*  
*method), 492* `get_ip_details()`  
`get_homing_params_block()` (*msl.equipment.resources.picotech.pt104.PT104*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*method), 526* `get_jog_mode()`  
`get_homing_velocity()` (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.KCubeStepperMotor*  
*(msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.KCubeStepperMotor*  
*method), 393* `get_jog_mode()`  
`get_homing_velocity()` (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors*  
*(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors*  
*method), 471* `get_jog_mode()`  
`get_homing_velocity()` (*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo*  
*method), 493* `get_jog_mode()`  
`get_homing_velocity()` (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*method), 526* `get_jog_params_block()`  
`get_hub_bay()` (*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo*  
*method), 493* `get_jog_params_block()`  
`get_hub_bay()` (*msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCuboSolenoid*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.KCuboSolenoid*  
*method), 516* `get_jog_params_block()`  
`get_hub_bay()` (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*method), 526* `get_jog_params_block()`  
`get_id()` (*msl.equipment.resources.thorlabs.fwx2c.FilterWhisker2C*  
*(msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo*  
*method), 580* `method), 493`  
`get_increment()` `get_jog_params_block()`  
*(msl.equipment.resources.optronic\_laboratories.ol756ocx.ol756ocx*  
*method), 204* *(msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor*  
*method), 527*  
`get_increment_index()` `get_jog_step_size()`  
*(msl.equipment.resources.optronic\_laboratories.ol756ocx.ol756ocx*  
*method), 204* *(msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.KCubeStepperMotor*  
*method), 393*  
`get_input_range()` `get_jog_step_size()`

([msl.equipment.resources.thorlabs.kinesis.integrated\\_stepper\\_motors.IntegratedStepperMotors](#)  
method), 471

[get\\_limit\\_switch\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_servo.KCubeDCServo](#)  
method), 493

[get\\_jog\\_step\\_size\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_stepper\\_motor.KCubeStepperMotor](#)  
method), 527

[get\\_jog\\_step\\_size\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.benchtop\\_stepper\\_motor.BenchtopStepperMotor](#)  
method), 394

[get\\_jog\\_vel\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_servo.KCubeDCServo](#)  
method), 493

[get\\_jog\\_vel\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.integrated\\_stepper\\_motors.IntegratedStepperMotors](#)  
method), 471

[get\\_jog\\_vel\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_servo.KCubeDCServo](#)  
method), 493

[get\\_jog\\_vel\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_stepper\\_motor.KCubeStepperMotor](#)  
method), 527

[get\\_joystick\\_params\(\)](#) ([msl.equipment.resources.mks\\_instruments.pr4000b.PR4000B](#)  
method), 145

[get\\_lambda\(\)](#) ([msl.equipment.resources.avantes.avaspec.AvaSpec](#)  
method), 114

[get\\_lamptime\(\)](#) ([msl.equipment.resources.energetiq.eq99.EQ99](#)  
method), 138

[get\\_led\\_switches\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.integrated\\_stepper\\_motors.IntegratedStepperMotors](#)  
method), 472

[get\\_led\\_switches\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_servo.KCubeDCServo](#)  
method), 494

[get\\_led\\_switches\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_solenoid.KCubeSolenoid](#)  
method), 516

[get\\_legacy\\_bus\\_scanning\(\)](#) ([msl.equipment.resources.nkt.nktpdll.NKT](#)  
static method), 169

[get\\_limit\\_mode\(\)](#) ([msl.equipment.resources.mks\\_instruments.pr4000b.PR4000B](#)  
method), 145

[get\\_limit\\_switch\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.benchtop\\_stepper\\_motor.BenchtopStepperMotor](#)  
method), 394

[get\\_limit\\_switch\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.integrated\\_stepper\\_motors.IntegratedStepperMotors](#)  
method), 472

[get\\_limit\\_switch\\_params\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_servo.KCubeDCServo](#)  
method), 494

[get\\_limit\\_switch\\_params\\_block\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.integrated\\_stepper\\_motors.IntegratedStepperMotors](#)  
method), 471

[get\\_limit\\_switch\\_params\\_block\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_servo.KCubeDCServo](#)  
method), 493

[get\\_linearization\\_point\(\)](#) ([msl.equipment.resources.mks\\_instruments.pr4000b.PR4000B](#)  
method), 145

[get\\_linearization\\_size\(\)](#) ([msl.equipment.resources.mks\\_instruments.pr4000b.PR4000B](#)  
method), 145

[get\\_lines\(\)](#) (in [msl.equipment.resources.utils](#) module)  
method), 82

[get\\_lines\(\)](#) ([msl.equipment.resources.utils.CHeader](#)  
method), 83

[get\\_log\(\)](#) ([msl.equipment.resources.bentham.benhw32.Bentham32](#)  
method), 123

[get\\_log\\_size\(\)](#) ([msl.equipment.resources.bentham.benhw32.Bentham32](#)  
method), 123

[get\\_lower\\_limit\(\)](#) ([msl.equipment.resources.mks\\_instruments.pr4000b.PR4000B](#)  
method), 145

[get\\_max\\_bw\(\)](#) ([msl.equipment.resources.bentham.benhw32.Bentham32](#)  
method), 123

[get\\_max\\_down\\_sample\\_ratio\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_a](#)  
method), 319

[get\\_max\\_ets\\_values\(\)](#) ([msl.equipment.resources.picotech.picoscope.ps3000a.Pic](#)  
method), 335

[get\\_max\\_segments\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_a](#)  
method), 319

[get\\_max\\_velocity\(\)](#) ([msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX](#)  
method), 589

[get\\_message\\_buffer\(\)](#) ([msl.equipment.resources.energetiq.eq99.EQ99](#)  
method), 138



|  |   |
|--|---|
| <code>get_min_step()</code>                      | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 371</code> |
| <code>get_min_velocity()</code>                  | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mmi_params()</code>                    | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 494</code> |
| <code>get_mmi_params_block()</code>              | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 494</code> |
| <code>get_mmi_params_block_ext()</code>          | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 528</code> |
| <code>get_modules()</code>                       | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 168</code> |
| <code>get_mono_backlash_steps()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 370</code> |
| <code>get_mono_detector_angle()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 370</code> |
| <code>get_mono_diverter_pos()</code>             | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 370</code> |
| <code>get_mono_diverter_pos_str()</code>         | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 371</code> |
| <code>get_mono_diverter_valid()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 371</code> |
| <code>get_mono_double()</code>                   | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 371</code> |
| <code>get_mono_double_intermediate_slit()</code> | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_double_subtractive()</code>       | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_exit_slit()</code>                | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_filter_max_pos()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_filter_min_pos()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_filter_position()</code>          | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_filter_present()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_focal_length()</code>             | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_gear_steps()</code>               | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating()</code>                  | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating_blaze()</code>            | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating_density()</code>          | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating_gadjust()</code>          | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating_installed()</code>        | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating_max()</code>              | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_grating_offset()</code>           | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |
| <code>get_mono_half_angle()</code>               | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.method), 372</code> |

|  |  |
|--|--|
| <code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 374</code>   | <code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 376</code>   |
| <code>get_mono_init_grating()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 374</code>         | <code>get_mono_slit_type_str()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 376</code>        |
| <code>get_mono_init_scan_rate_nm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 374</code>    | <code>get_mono_slit_width()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 377</code>           |
| <code>get_mono_init_wave_nm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 374</code>         | <code>get_mono_slit_width_max()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 377</code>       |
| <code>get_mono_int_led_on()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>           | <code>get_mono_turret()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 377</code>               |
| <code>get_mono_items()</code><br><code>(msl.equipment.resources.bentham.benhw32.Bentham32.method), 122</code>  | <code>get_mono_turret_gratings()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>      |
| <code>get_mono_items()</code><br><code>(msl.equipment.resources.bentham.benhw64.Bentham64.method), 124</code>  | <code>get_mono_turret_max()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 377</code>           |
| <code>get_mono_model()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>                | <code>get_mono_wavelength_abs_cm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>    |
| <code>get_mono_motor_int()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>            | <code>get_mono_wavelength_ang()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>       |
| <code>get_mono_nm_rev_ratio()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 379</code>         | <code>get_mono_wavelength_cutoff_nm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code> |
| <code>get_mono_precision()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>            | <code>get_mono_wavelength_ev()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>        |
| <code>get_mono_preopen_model()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.static method), 379</code> | <code>get_mono_wavelength_micron()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>    |
| <code>get_mono_scan_rate_nm_min()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>     | <code>get_mono_wavelength_min_nm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>    |
| <code>get_mono_serial()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>               | <code>get_mono_wavelength_nm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 378</code>        |
| <code>get_mono_shutter_open()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>         | <code>get_mono_wavelength_rel_cm()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 379</code>    |
| <code>get_mono_shutter_valid()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>        | <code>get_mono_wheel_int()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 379</code>            |
| <code>get_mono_sine_drive()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 375</code>           | <code>get_motor_params()</code><br><code>(msl.equipment.resources.princeton_instruments.arc_instrument.princeton_instruments.arc_instrument.method), 395</code>              |
| <code>get_mono_slit_type()</code>  | <code>get_motor_params()</code>  |

(msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 472

get\_motor\_params() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 495

get\_motor\_params() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 528

get\_motor\_params\_ext() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 395

get\_motor\_params\_ext() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 472

get\_motor\_params\_ext() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 495

get\_motor\_params\_ext() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 529

get\_motor\_travel\_limits() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 395

get\_motor\_travel\_limits() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 473

get\_motor\_travel\_limits() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 496

get\_motor\_travel\_limits() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 530

get\_motor\_travel\_limits() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 529

get\_motor\_travel\_mode() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 395

get\_motor\_travel\_mode() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 473

get\_motor\_travel\_mode() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 496

get\_motor\_travel\_mode() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 529

get\_motor\_velocity\_limits() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 396

get\_motor\_velocity\_limits() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 473

get\_motor\_velocity\_limits() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 496

get\_motor\_velocity\_limits() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 529

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 396

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 473

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 496

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 529

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 396

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 473

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 496

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 530

get\_move\_absolute\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 123

get\_ncl\_filter\_max\_pos() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 379

get\_ncl\_filter\_min\_pos() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 379

get\_ncl\_filter\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 379

get\_ncl\_filter\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 379

get\_ncl\_filter\_present() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motors.kcube\_stepper\_motors  
 method), 379

get\_ncl\_mono\_enum() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.benchtop\_stepper\_motors  
 method), 380

get\_ncl\_mono\_setup() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.integrated\_stepper\_motors.kcube\_dc\_servo  
 method), 380

get\_ncl\_shutter\_open() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.kcube\_dc\_servo  
 method), 380

(msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 method), 380 get\_num\_pixels() (msl.equipment.resources.avantes.avaspec.Avantes  
 (msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 method), 380 get\_number\_positions() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor, 467  
 get\_ncl\_shutter\_valid() (msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 method), 380 get\_number\_positions() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.Flipper  
 get\_ncl\_ttl\_in() (msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 method), 380 get\_number\_positions() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motor, 467  
 get\_ncl\_ttl\_out() (msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 method), 380 get\_number\_positions() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo, 496  
 get\_ncl\_ttl\_valid() (msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 method), 380 get\_number\_positions() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor, 467  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor, 467 BenchtopStepperMotor  
 method), 396 get\_number\_positions() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor, 467  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.Flipper  
 method), 467 get\_ocx\_version() (msl.equipment.resources.optronic\_laboratories.ol756ocx, 405  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motor, 467 IntegratedStepperMotors  
 method), 474 get\_oem\_parameter() (msl.equipment.resources.avantes.avaspec.Avantes  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo, 496 KCubeDCServo  
 method), 496 get\_offset() (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b, 146  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo, 496 KCubeDCServo  
 method), 517 get\_open\_id() (msl.equipment.resources.nkt.nktpdll.NKT  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo, 496 KCubeDCServo  
 static method), 169 get\_operating\_mode() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid, 517  
 get\_next\_message() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo, 496 KCubeDCServo  
 method), 530 get\_operating\_mode() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid, 517  
 get\_no\_of\_captures() (msl.equipment.resources.picotech.picoscope.picoscope.Api  
 method), 319 get\_operating\_mode() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid, 517  
 get\_no\_of\_dark\_currents() (msl.equipment.resources.bentham.benhwa32.Benthump32, 122  
 method), 122 get\_output() (msl.equipment.resources.energetiq.eq99.EQ99  
 get\_num\_channels() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor, 467 BenchtopStepperMotor  
 method), 397 get\_output\_range() (msl.equipment.resources.princeton\_instruments.pr4000b.PR4000b, 146  
 get\_num\_det() (msl.equipment.resources.princeton\_instruments.pr4000b.PR4000b, 146  
 method), 381 get\_over\_current\_protection() (msl.equipment.resources.aim\_tti.mx\_series.MXSeries  
 get\_num\_devices() (msl.equipment.resources.avantes.avaspec.Avantes (msl.equipment.resources.aim\_tti.mx\_series.MXSeries  
 method), 114 method), 85  
 get\_num\_found\_inst\_ports() (msl.equipment.resources.princeton\_instruments.arc\_nishoudy, 320 PrincetonInstruments  
 static method), 381 get\_over\_voltage\_protection() (msl.equipment.resources.aim\_tti.mx\_series.MXSeries  
 method), 86  
 get\_num\_of\_processed\_captures() get\_parameter() (msl.equipment.resources.picotech.picoscope.picoscope.Api  
 (msl.equipment.resources.picotech.picoscope.picoscope.picoscope.Api (msl.equipment.resources.avantes.avaspec.Avantes



---

|              |            |
|--------------|------------|
| <b>Index</b> | <b>641</b> |
|--------------|------------|

|   |  |
|---|--|
| <code>get_readout_serial()</code><br>( <code>msl.equipment.resources.princeton_instruments.arc_1500b.Arc1500B</code><br>method), 381                                    | <code>get_setpoint()</code><br>( <code>msl.equipment.princeton_instruments.raicol_tec.RaicolTEC</code><br>method), 388                                       |
| <code>get_real_value_from_device_unit()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.BenchtopStepperMotors</code><br>method), 398 | <code>get_settling_time()</code><br>( <code>msl.equipment.princeton_instruments.raicol_tec.RaicolTEC</code><br>method), 206                                  |
| <code>get_real_value_from_device_unit()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.integrated_steppers.IntegratedStepperMotors</code><br>method), 475   | <code>get_setup()</code> ( <code>msl.equipment.resources.optronic_laboratories.ol756cx</code><br>method), 216  |
| <code>get_real_value_from_device_unit()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo</code><br>method), 497                   | <code>get_shutter_init()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 147  |
| <code>get_real_value_from_device_unit()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor</code><br>method), 531         | <code>get_shutter_state()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 147   |
| <code>get_relays()</code> ( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 146  | <code>get_signal_array()</code><br>( <code>msl.equipment.resources.optronic_laboratories.ol756cx</code><br>method), 206                                      |
| <code>get_remote_mode()</code><br>( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 146  | <code>get_signal_mode()</code><br>( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 147                                     |
| <code>get_remote_mode_error()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 140  | <code>get_soft_limit_mode()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.BenchtopStepperMotors</code><br>method), 398  |
| <code>get_resolution()</code><br>( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 146   | <code>get_soft_limit_mode()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.integrated_steppers.IntegratedStepperMotors</code><br>method), 475    |
| <code>get_rtd_offset()</code><br>( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 146   | <code>get_soft_limit_mode()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo</code><br>method), 497                    |
| <code>get_sasl_mechanism_list()</code><br>( <code>msl.equipment.hislip.SyncClient</code><br>method), 70   | <code>get_soft_limit_mode()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor</code><br>method), 531          |
| <code>get_saturated_pixels()</code><br>( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>method), 114   | <code>get_software_version()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.BenchtopStepperMotors</code><br>method), 399 |
| <code>get_scale()</code> ( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 147   | <code>get_software_version()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper</code><br>method), 464                  |
| <code>get_scan_mode()</code><br>( <code>msl.equipment.resources.optronic_laboratories.ol756cx</code><br>method), 205  | <code>get_software_version()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.integrated_steppers.IntegratedStepperMotors</code><br>method), 475   |
| <code>get_sensor()</code> ( <code>msl.equipment.resources.electron_dynamics.tc_series.TCSeries</code><br>method), 131   | <code>get_software_version()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo</code><br>method), 497                   |
| <code>get_sensor_mode()</code><br>( <code>msl.equipment.resources.thorlabs.fwx2c.FilterWheelX2C</code><br>method), 580  | <code>get_software_version()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KCubeSolenoid</code><br>method), 517                  |
| <code>get_setpoint()</code><br>( <code>msl.equipment.resources.electron_dynamics.tc_series.TCSeries</code><br>method), 131  | <code>get_software_version()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor</code><br>method), 531         |
| <code>get_setpoint()</code><br>( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 147   | <code>get_solenoid_state()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KCubeSolenoid</code><br>method), 517                    |

`(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KCubeDCServer`  
`method), 517` `(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KCubeDCServer`  
`get_speed()` `(msl.equipment.resources.optosigma.shot702.SHOT702`, 498  
`method), 191` `get_status_bits()`  
`get_speed_mode()` `(msl.equipment.resources.thorlabs.kinesis.kcube_solenoid`  
`(msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX2C`, 517  
`method), 580` `get_status_bits()`  
`get_speed_origin()` `(msl.equipment.resources.thorlabs.kinesis.kcube_stepper`  
`(msl.equipment.resources.optosigma.shot702.SHOT702`, 532  
`method), 191` `get_status_by_serial()`  
`get_stage_axis_max_pos()` `(msl.equipment.resources.avantes.avaspec.Avantes`  
`(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor`  
`method), 399` `get_std_file_enabled()`  
`get_stage_axis_max_pos()` `(msl.equipment.resources.optronic_laboratories.ol756cx`  
`(msl.equipment.resources.thorlabs.kinesis.integrated_steppers.IntegratedStepperMotors`  
`method), 475` `get_steps()` `(msl.equipment.resources.optosigma.shot702.SHOT`  
`method), 191`  
`get_stage_axis_max_pos()` `(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KCubeDCServer`  
`method), 498` `method), 123`  
`get_stage_axis_max_pos()` `get_streaming_last_values()`  
`(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor`  
`method), 531` `method), 316`  
`get_stage_axis_min_pos()` `get_streaming_latest_values()`  
`(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor`  
`method), 399` `method), 320`  
`get_stage_axis_min_pos()` `get_streaming_values()`  
`(msl.equipment.resources.thorlabs.kinesis.integrated_steppers.IntegratedStepperMotors`  
`method), 476` `method), 316`  
`get_stage_axis_min_pos()` `get_streaming_values_no_aggregation()`  
`(msl.equipment.resources.thorlabs.kinesis.kcube_dc_server.KCubeDCServer`  
`method), 498` `method), 316`  
`get_stage_axis_min_pos()` `get_string()` `(msl.equipment.resources.picotech.picoscope.picoscope_2`  
`(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor`  
`method), 532` `method), 316`  
`get_standard_file()` `get_struct_imports()`  
`(msl.equipment.resources.optronic_laboratories.ol756cx`, 206  
`method), 206`  
`get_start_wavelength()` `get_termination()`  
`(msl.equipment.resources.optronic_laboratories.ol756cx`, 206  
`method), 206` `(msl.equipment.resources.energetiq.eq99.EQ99`  
`method), 206` `get_text_between_brackets()`  
`get_status()` `(msl.equipment.resources.electron_dynamics.femto_sis.FemtoSis`  
`method), 131` `static method), 83`  
`get_status_bits()` `get_time_to_current_pos()`  
`(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor`  
`method), 399` `method), 580`  
`get_status_bits()` `get_timebase()`  
`(msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper`  
`method), 466` `method), 317`  
`get_status_bits()` `get_timebase()`  
`(msl.equipment.resources.thorlabs.kinesis.integrated_steppers.IntegratedStepperMotors`  
`method), 476` `method), 320`

|  |   |
|--|---|
| <code>get_timebase2()</code>   | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.picoscope_api</code> | <code>method), 320</code>   |
| <code>get_timeout()</code>   | <code>(msl.equipment.hislip.HiSLIPClient</code>                         |
| <code>method), 68</code>   | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper</code>    |
| <code>get_times_and_values()</code>                                    | <code>method), 532</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.picoscope_api</code> | <code>method), 317</code>   |
| <code>get_transit_time()</code>  | <code>get_trigger_params_params_block()</code>                          |
| <code>(msl.equipment.resources.thorlabs.kinesis.filter_flipper</code>  | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code>   |
| <code>method), 465</code>  | <code>method), 499</code>   |
| <code>get_travel_per_pulse()</code>                                    | <code>get_trigger_params_params_block()</code>                          |
| <code>(msl.equipment.resources.optosigma.shot702.SHOT702</code>        | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper</code>    |
| <code>method), 191</code>  | <code>method), 533</code>   |
| <code>get_trigger_channel_time_offset()</code>                         | <code>get_trigger_switches()</code>                                     |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000</code>        | <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepp</code>   |
| <code>method), 337</code>  | <code>method), 399</code>   |
| <code>get_trigger_channel_time_offset64()</code>                       | <code>get_trigger_switches()</code>                                     |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000</code>        | <code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepp</code> |
| <code>method), 337</code>  | <code>method), 476</code>   |
| <code>get_trigger_config_params()</code>                               | <code>get_trigger_time_offset()</code>                                  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code>  | <code>(msl.equipment.resources.picotech.picoscope.picoscope_a</code>    |
| <code>method), 498</code>  | <code>method), 320</code>   |
| <code>get_trigger_config_params()</code>                               | <code>get_trigger_time_offset64()</code>                                |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_sol</code>       | <code>(msl.equipment.resources.picotech.picoscope.picoscope_a</code>    |
| <code>method), 518</code>  | <code>method), 320</code>   |
| <code>get_trigger_config_params()</code>                               | <code>get_unit_info()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper</code>   | <code>(msl.equipment.resources.picotech.picoscope.picoscope.P</code>    |
| <code>method), 532</code>  | <code>method), 313</code>   |
| <code>get_trigger_config_params_block()</code>                         | <code>get_unit_info()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code>  | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper</code>    |
| <code>method), 498</code>  | <code>method), 359</code>   |
| <code>get_trigger_config_params_block()</code>                         | <code>get_upper_limit()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_sol</code>       | <code>(msl.equipment.resources.mks_instruments.pr4000b.PR40</code>      |
| <code>method), 518</code>  | <code>method), 147</code>   |
| <code>get_trigger_config_params_block()</code>                         | <code>get_value()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_sol</code>       | <code>(msl.equipment.resources.picotech.pt104.PT104</code>              |
| <code>method), 518</code>  | <code>method), 147</code>   |
| <code>get_trigger_config_params_block()</code>                         | <code>get_values()</code>   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code>  | <code>(msl.equipment.resources.picotech.picoscope.picos</code>          |
| <code>method), 532</code>  | <code>method), 317</code>   |
| <code>get_trigger_info_bulk()</code>                                   | <code>get_values_async()</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.ps3000</code>        | <code>(msl.equipment.resources.picotech.picoscope.picos</code>          |
| <code>method), 335</code>  | <code>method), 321</code>   |
| <code>get_trigger_info_bulk()</code>                                   | <code>get_values_bulk()</code>  |
| <code>(msl.equipment.resources.picotech.picoscope.ps5000</code>        | <code>(msl.equipment.resources.picotech.picoscope.picos</code>          |
| <code>method), 343</code>  | <code>method), 321</code>   |
| <code>get_trigger_mode()</code>  | <code>get_values_bulk_async()</code>                                    |
| <code>(msl.equipment.resources.energetiq.eq99.EQ99</code>              | <code>(msl.equipment.resources.picotech.picoscope.ps6000.Pico</code>    |
| <code>method), 139</code>  | <code>method), 344</code>   |
| <code>get_trigger_mode()</code>  | <code>get_values_overlapped()</code>                                    |
| <code>(msl.equipment.resources.thorlabs.fwx2c.FilterWheel</code>       | <code>(msl.equipment.resources.picotech.picoscope.picoscope_a</code>    |
| <code>method), 581</code>  | <code>method), 321</code>   |
| <code>get_trigger_params_params()</code>                               | <code>get_values_overlapped_bulk()</code>                               |



(msl.equipment.resources.picotech.picoscope.get\_values\_page\_size() method), 322  
 (msl.equipment.resources.aim\_tti.mx\_series.MXSeries get\_values\_trigger\_channel\_time\_offset\_bulk() method), 86  
 (msl.equipment.resources.picotech.picoscope.get\_4000\_page\_size\_4000() method), 337  
 (msl.equipment.resources.aim\_tti.mx\_series.MXSeries get\_values\_trigger\_channel\_time\_offset\_bulk64() method), 86  
 (msl.equipment.resources.picotech.picoscope.get\_4000\_page\_size\_64() method), 337  
 (msl.equipment.resources.aim\_tti.mx\_series.MXSeries get\_values\_trigger\_time\_offset\_bulk() method), 86  
 (msl.equipment.resources.picotech.picoscope.get\_values\_page\_size() method), 322  
 (msl.equipment.resources.aim\_tti.mx\_series.MXSeries get\_values\_trigger\_time\_offset\_bulk64() method), 86  
 (msl.equipment.resources.picotech.picoscope.get\_valves() method), 322  
 (msl.equipment.resources.mks\_instrument.get\_valves() method), 147  
 (msl.equipment.resources.bentham.benhw32.Bentham32 get\_val\_params() method), 122  
 (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor get\_val\_params() method), 400  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors get\_val\_params() method), 476  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_stepper\_motor.KCubeDCStepperMotor get\_val\_params() method), 499  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor get\_val\_params\_block() method), 533  
 (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor get\_val\_params\_block() method), 400  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors get\_val\_params\_block() method), 476  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_stepper\_motor.KCubeDCStepperMotor get\_val\_params\_block() method), 499  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor get\_val\_params\_block() method), 533  
 (msl.equipment.resources.bentham.greater\_bham.get\_version() method), 123  
 (msl.equipment.resources.optosigma.get\_version() method), 191  
 (msl.equipment.resources.picotech.picoscope.get\_version\_info() method), 114  
 (msl.equipment.resources.aim\_tti.mx\_series.MXSeries get\_voltage() method), 86  
 (msl.equipment.resources.optronic\_laboratories.get\_voltage() method), 217

attribute), 268  
 GREATER\_THAN (msl.equipment.resources.picotech.picoscope.enums.PS5000a.PulseWidthType attribute), 301  
 GREATER\_THAN (msl.equipment.resources.picotech.picoscope.enums.PS5000a.PulseWidthType attribute), 292  
 GREATER\_THAN (msl.equipment.resources.picotech.picoscope.enums.PS6000a.PulseWidthType attribute), 310  
 GREEN (msl.equipment.resources.picotech.picoscope.enums.PS4000a.ChannelLed attribute), 275  
 group\_add() (msl.equipment.resources.bentham.benhw32.Bentham32.Device attribute), 122  
 group\_execute\_trigger() (msl.equipment.connection\_prologix.ConnectionPrologix.Device attribute), 28  
 group\_remove() (msl.equipment.resources.bentham.benhw32.Bentham32.Device attribute), 122

## H

H (msl.equipment.resources.picotech.picoscope.enums.PS4000a.Channel attribute), 271  
 H\_MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000a.ChannelBufferIndex attribute), 272  
 H\_MIN (msl.equipment.resources.picotech.picoscope.enums.PS4000a.ChannelBufferIndex attribute), 272  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000a.WaveType attribute), 237  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000a.WaveType attribute), 229  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS4000a.WaveType attribute), 253  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS4000a.WaveType attribute), 275  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS4000a.WaveType attribute), 265  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000a.WaveType attribute), 297  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000a.WaveType attribute), 288  
 HALF\_SINE (msl.equipment.resources.picotech.picoscope.enums.PS6000a.WaveType attribute), 306  
 HALF\_SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000a attribute), 332  
 HALF\_SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000a attribute), 335  
 HALF\_SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000a attribute), 339  
 HALF\_SINE\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.PicoScope4000a attribute), 340  
 handle (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope attribute), 240  
 HARDWARE\_VERSION (msl.equipment.resources.picotech.picoscope.enums.PicoScope4000a attribute), 340

attribute), 224

**HARDWARE\_VERSION**  
(*msl.equipment.resources.picotech.picoscope.enums.PS2000.Information* attribute), 226

**HARDWARE\_VERSION**  
(*msl.equipment.resources.picotech.picoscope.enums.PS2000.Information* attribute), 244

**hardwareVersion**  
(*msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Joystick.Parameters* attribute), 556

**has\_last\_msg\_timer\_overrun()**  
(*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor* method), 400

**has\_last\_msg\_timer\_overrun()**  
(*msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper* method), 467

**has\_last\_msg\_timer\_overrun()**  
(*msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors* method), 476

**has\_last\_msg\_timer\_overrun()**  
(*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo* method), 499

**has\_last\_msg\_timer\_overrun()**  
(*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor* method), 518

**has\_last\_msg\_timer\_overrun()**  
(*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor* method), 533

**header** (*msl.equipment.hislip.Message* attribute), 53

**heartbeat()** (*msl.equipment.resources.avantes.avaspec.Avantes* method), 115

**HeartbeatRespType** (class in *HubAnalogueModes* (*msl.equipment.resources.avantes.avaspec*), 98

**HIGH** (*msl.equipment.resources.picotech.picoscope.enums.PS2000.Information* attribute), 240

**HIGH** (*msl.equipment.resources.picotech.picoscope.enums.PS2000.Information* attribute), 257

**highGearAcceleration**  
(*msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Joystick.Parameters* attribute), 558

**highGearMaxVelocity**  
(*msl.equipment.resources.thorlabs.kinesis.structs.MQTT\_Joystick.Parameters* attribute), 558

**highVoltageOutputRoute**  
(*msl.equipment.resources.thorlabs.kinesis.structs.KNA\_ArduinoSettings* attribute), 571

**highVoltageOutRange**  
(*msl.equipment.resources.thorlabs.kinesis.structs.KNA\_ArduinoSettings* attribute), 571

**HiSLIPClient** (class in *msl.equipment.hislip*), 67

**HiSLIPException**, 52

**home()** (*msl.equipment.resources.optosigma.shot702.SHOT702* method), 322

**home()** (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor* method), 401

**home()** (*msl.equipment.resources.thorlabs.kinesis.filter\_flipper.FilterFlipper* method), 464

**home()** (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors* method), 476

**home()** (*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo* method), 500

**home()** (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor* method), 534

**horizontalComponent**

**host** (*msl.equipment.connection\_socket.ConnectionSocket* property), 37

**host** (*msl.equipment.connection\_tcpip\_hislip.ConnectionTCPIPHislip* property), 37

**host** (*msl.equipment.connection\_tcpip\_vxi11.ConnectionTCPIPVXI11* property), 40

**host** (*msl.equipment.connection\_zeromq.ConnectionZeroMQ* property), 40

**Hour** (*msl.equipment.resources.nkt.nktpdll.DateTimeType* attribute), 153

**hubAnalogOutput**

**hubAnalogOutput** (*msl.equipment.resources.thorlabs.kinesis.structs.TSG\_IO* attribute), 576

**HubAnalogueModes** (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 452

**home** (*msl.equipment.resources.omega.itx.iTHX* method), 187

**HW\_TRIGGER\_MODE**

**hysteresis** (*msl.equipment.resources.picotech.picoscope.structs.PS2000.Information* attribute), 347

**hysteresis** (*msl.equipment.resources.picotech.picoscope.structs.PS2000.Information* attribute), 354

**hysteresisLower**  
(*msl.equipment.resources.picotech.picoscope.structs.PS2000.Information* attribute), 357





attribute), 290  
 INVALID\_ID\_INIT\_SEQUENCE  
 INSIDE (msl.equipment.resources.picotech.picoscope.enums.PS6000TypesHoldErrorType attribute), 309  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 560  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 559  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 561  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 566  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 573  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 573  
 integralGain (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 578  
 integralLimit (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 560  
 integralLimit (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 559  
 integralLimit (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 561  
 integralTerm (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 572  
 integralTerm (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 564  
 IntegratedStepperMotors (class in msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters), 322  
 is\_calibrated (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters attribute), 468  
 IntegrationTime (class in msl.equipment.resources.thorlabs.kinesis.structs.MOT\_DC\_PIDParameters), 125  
 Interface (class in msl.equipment.constants), 44  
 interface (msl.equipment.record\_types.ConnectionRecord attribute), 79  
 InterfaceType (class in msl.equipment.constants), 87  
 InterfaceType (msl.equipment.resources.avantes.avaspec.BroadbandSpectrumAnalyzerType attribute), 93  
 interrupt\_handler() (msl.equipment.vxi11.RPCClient method), 589  
 Interrupted (msl.equipment.hislip.MessageType attribute), 51  
 INVALID\_AV5\_HANDLE\_VALUE (msl.equipment.resources.avantes.avaspec.Avantes attribute), 100

attribute), 555

iTHX (class in `msl.equipment.resources.omega.ithx`), 186

## J

jerk (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_ValabineProfileParameters attribute), 557

Jog (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_ButtonMade attribute), 460

JogContinuous (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_JogMode attribute), 439

JogContinuous (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_JogMode attribute), 459

JogStep (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_JogMode attribute), 439

JogStep (msl.equipment.resources.thorlabs.kinesis.enums.TIM\_JogMode attribute), 459

Joystick (msl.equipment.resources.thorlabs.kinesis.enums.PPG\_IOControlMode attribute), 434

JoystickBnc (msl.equipment.resources.thorlabs.kinesis.enums.PPG\_IOControlMode attribute), 434

JoystickDirectionSense (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MMIPParams attribute), 572

JoystickMode (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MMIPParams attribute), 572

JS\_GotoPosition (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_JoystickModes attribute), 442

JS\_Jog (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_JoystickModes attribute), 442

JS\_Velocity (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_JoystickModes attribute), 442

## K

KCube\_Brushless\_Motor (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 552

KCube\_DC\_Servo (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 552

KCube\_Inertial\_Motor (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 552

KCube\_LaserSource (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 552

KCube\_NanoTrak (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 552

KCube\_Piezo (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl attribute), 552

KCube\_Solenoid (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 552

KCube\_Stepper\_Motor (msl.equipment.resources.thorlabs.kinesis.motion\_control attribute), 552

KCubeDCServo (class in msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo), 490

KCubeSolenoid (class in msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid), 490

KCubeStepperMotor (class in msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_r), 523

KD\_TrigOut\_GPO (msl.equipment.resources.thorlabs.kinesis.enums.QD\_KP attribute), 454

KD\_TrigOut\_Sum (msl.equipment.resources.thorlabs.kinesis.enums.QD\_KP attribute), 454

KernelDriverVersion (msl.equipment.resources.picotech.picoscope.enums.PS20 attribute), 244

KernelVersion (msl.equipment.resources.picotech.picoscope.enums.Pico attribute), 224

KIM\_channels (class in msl.equipment.resources.thorlabs.kinesis.enums), 438

KIM\_DirectionSense (class in msl.equipment.resources.thorlabs.kinesis.enums), 440

KIM\_DriveParameters (class in msl.equipment.resources.thorlabs.kinesis.structs), 568

KIM\_FBSignalMode (class in msl.equipment.resources.thorlabs.kinesis.enums), 439

KIM\_FeedbackSigParams (class in msl.equipment.resources.thorlabs.kinesis.structs), 569

**KIM\_HomeParameters** (class in attribute), 444  
*msl.equipment.resources.thorlabs.kinesis.structs*, 568  
**KIM\_LaserOn** (*msl.equipment.resources.thorlabs.kinesis.enums.KLD\_TriggerMode* attribute), 444  
**KIM\_JogMode** (class in *KLD\_LowStability* *msl.equipment.resources.thorlabs.kinesis.enums*), (class in *KLD\_TriggerMode* attribute), 444  
**KIM\_JogParameters** (class in *KLD\_MMIParams* (class in *msl.equipment.resources.thorlabs.kinesis.structs*), (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 568  
**KIM\_JoystickModes** (class in *KLD\_Output* (*msl.equipment.resources.thorlabs.kinesis.enums.KLD\_SetPointChange* attribute), 444  
**KIM\_LimitSwitchModes** (class in (class in *msl.equipment.resources.thorlabs.kinesis.enums.KLD\_TriggerMode* attribute), 444  
**KIM\_LimitSwitchParameters** (class in *msl.equipment.resources.thorlabs.kinesis.enums*), (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 444  
**KIM\_MMIOChannelParameters** (class in *KLD\_TriggerIOParams* (class in *msl.equipment.resources.thorlabs.kinesis.structs*), (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 569  
**KIM\_MMIParameters** (class in *KLD\_TriggerPol\_High* (*msl.equipment.resources.thorlabs.kinesis.enums.KLD\_TriggerMode* attribute), 445  
**KIM\_Status** (class in *KLD\_TriggerPol\_Low* (*msl.equipment.resources.thorlabs.kinesis.enums.KLD\_TriggerMode* attribute), 445  
**KIM\_TravelDirection** (class in *KLD\_TriggerPolarity* (class in *msl.equipment.resources.thorlabs.kinesis.enums*), (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 444  
**KIM\_TriggerIOConfig** (class in *KLS\_ConstantCurrent* (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_OperationMode* attribute), 445  
**KIM\_TriggerModes** (class in *KLS\_ConstantPower* (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_OperationMode* attribute), 445  
**KIM\_TriggerParamsParameters** (class in *KLS\_Disabled* (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_OperationMode* attribute), 445  
**KIM\_TriggerPolarities** (class in *KLS\_HighStability* (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_TriggerMode* attribute), 446  
**KLD\_Disabled** (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_Input* attribute), 444  
**KLD\_HighStability** (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_InterlockEnabledMode* attribute), 446  
**KLD\_Input** (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_KLS\_OperationMode* attribute), 444  
**KLD\_InterlockEnabled** (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_KLS\_OperationMode* attribute), 446  
**KLS\_MMIParams** (class in *KLS\_MMIParams* (class in *msl.equipment.resources.thorlabs.kinesis.structs*), (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 569

*msl.equipment.resources.thorlabs.kinesis.structs*), *attribute*), 438  
569 KMOT\_TrigIn\_RelativeMove  
KLS\_OpMode (class in *msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigIn\_RelativeMove*  
*msl.equipment.resources.thorlabs.kinesis.enums*), *attribute*), 438  
445 KMOT\_TrigOut\_AtMaxVelocity  
KLS\_Output (*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_TriggerMode* resources.thorlabs.kinesis.enums.KMOT\_TrigOut\_AtMaxVelocity  
*attribute*), 445 *attribute*), 438  
KLS\_SetPointChange KMOT\_TrigOut\_AtPositionSteps  
(*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_SetPointChange* resources.thorlabs.kinesis.enums.KMOT\_TrigOut\_AtPositionSteps  
*attribute*), 446 *attribute*), 438  
KLS\_TriggerMode (class in KMOT\_TrigOut\_GPO  
*msl.equipment.resources.thorlabs.kinesis.enums*), (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigOut\_GPO*  
445 *attribute*), 438  
KLS\_TrigIOParams (class in KMOT\_TrigOut\_InMotion  
*msl.equipment.resources.thorlabs.kinesis.structs*), (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigOut\_InMotion*  
570 *attribute*), 438  
KLS\_TrigPol\_High KMOT\_TrigOut\_Synch  
(*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_TrigPol\_High* resources.thorlabs.kinesis.enums.KMOT\_TrigOut\_Synch  
*attribute*), 446 *attribute*), 438  
KLS\_TrigPol\_Low KMOT\_TrigPolarityHigh  
(*msl.equipment.resources.thorlabs.kinesis.enums.KLS\_TrigPol\_Low* resources.thorlabs.kinesis.enums.KMOT\_TrigPolarityHigh  
*attribute*), 446 *attribute*), 438  
KLS\_TrigPolarity (class in KMOT\_TrigPolarityLow  
*msl.equipment.resources.thorlabs.kinesis.enums*), (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigPolarityLow*  
446 *attribute*), 438  
KMOT\_MMIParams (class in KMOT\_WheelDirectionSense (class in  
*msl.equipment.resources.thorlabs.kinesis.structs*), *msl.equipment.resources.thorlabs.kinesis.enums*),  
567 437  
KMOT\_TrigDisabled KMOT\_WheelMode (class in  
(*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigDisabled* resources.thorlabs.kinesis.enums),  
*attribute*), 438 437  
KMOT\_TriggerConfig (class in KMOT\_WM\_Jog (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_WM\_Jog*  
*msl.equipment.resources.thorlabs.kinesis.structs*), *attribute*), 437  
567 KMOT\_WM\_MoveAbsolute  
KMOT\_TriggerParams (class in (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerParams*  
*msl.equipment.resources.thorlabs.kinesis.structs*), *attribute*), 438  
568 KMOT\_WM\_Negative  
KMOT\_TriggerPortMode (class in (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerPortMode*  
*msl.equipment.resources.thorlabs.kinesis.enums*), *attribute*), 437  
438 KMOT\_WM\_Positive  
KMOT\_TriggerPortPolarity (class in (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TriggerPortPolarity*  
*msl.equipment.resources.thorlabs.kinesis.enums*), *attribute*), 437  
438 KMOT\_WM\_Velocity  
KMOT\_TrigIn\_AbsoluteMove (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigIn\_AbsoluteMove*  
(*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigIn\_AbsoluteMove* *attribute*), 438  
KMOT\_TrigIn\_GPI KNA\_Channel1 (*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigIn\_GPI*  
(*msl.equipment.resources.thorlabs.kinesis.enums.KNA\_Channel1* resources.thorlabs.kinesis.enums.KNA\_Channel1  
*attribute*), 438 *attribute*), 450  
KMOT\_TrigIn\_Home KNA\_Channels (class in  
(*msl.equipment.resources.thorlabs.kinesis.enums.KMOT\_TrigIn\_Home* resources.thorlabs.kinesis.enums),  
*attribute*), 438 *attribute*), 450



|  |  |
|--|--|
| 450  | 571  |
| KNA_ChannelUndefined   | KNA_TTIARange (class in  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_ChannelUndefined | (msl.equipment.resources.thorlabs.kinesis.enums),                      |
| attribute), 450  | 446  |
| KNA_Default_Range  | KNA_TTIARange10_166uA  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_Default_Range    | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange10_166uA  |
| attribute), 448  | attribute), 447  |
| KNA_Default_Route  | KNA_TTIARange11_500uA  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_Default_Route    | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange11_500uA  |
| attribute), 448  | attribute), 447  |
| KNA_EnableInputBoost   | KNA_TTIARange12_1_66mA   |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_EnableInputBoost | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange12_1_66mA |
| attribute), 448  | attribute), 447  |
| KNA_ExtIn_IO1  | KNA_TTIARange13_500uA  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_ExtIn_IO1        | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange13_500uA  |
| attribute), 448  | attribute), 447  |
| KNA_ExtIn_PIN  | KNA_TTIARange14_500uA  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_ExtIn_PIN        | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange14_500uA  |
| attribute), 448  | attribute), 447  |
| KNA_ExtOut_Dis   | KNA_TTIARange15_500uA  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_ExtOut_Dis       | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange15_500uA  |
| attribute), 448  | attribute), 447  |
| KNA_ExtOut_IO2   | KNA_TTIARange16_500uA  |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_ExtOut_IO2       | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange16_500uA  |
| attribute), 448  | attribute), 447  |
| KNA_FeedbackLoopConstants  | KNA_TTIARange17_500uA  |
| (class in msl.equipment.resources.thorlabs.kinesis.structs),         | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange17_500uA  |
| 571  | attribute), 447  |
| KNA_FeedbackModeTypes  | KNA_TTIARange18_500uA  |
| (class in msl.equipment.resources.thorlabs.kinesis.enums),           | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange18_500uA  |
| 450  | attribute), 447  |
| KNA_FeedbackSource   | KNA_TTIARange19_500uA  |
| (class in msl.equipment.resources.thorlabs.kinesis.enums),           | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange19_500uA  |
| 446  | attribute), 447  |
| KNA_HighOutputVoltageRoute   | KNA_TTIARange20_1_66uA   |
| (class in msl.equipment.resources.thorlabs.kinesis.enums),           | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange20_1_66uA |
| 448  | attribute), 447  |
| KNA_HighVoltageRange   | KNA_TTIARange21_5uA  |
| (class in msl.equipment.resources.thorlabs.kinesis.enums),           | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange21_5uA    |
| 447  | attribute), 447  |
| KNA_IO1Only  | KNA_TTIARange22_16_6uA   |
| (msl.equipment.resources.thorlabs.kinesis.enums.KNA_IO1Only          | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange22_16_6uA |
| attribute), 447  | attribute), 447  |
| KNA_IOSettings   | KNA_TTIARange23_50uA   |
| (class in msl.equipment.resources.thorlabs.kinesis.structs),         | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TTIARange23_50uA   |
| 571  | attribute), 447  |
| KNA_LowOutputVoltageRoute  | KNA_TTIARangeParameters  |
| (class in msl.equipment.resources.thorlabs.kinesis.enums),           | (class in msl.equipment.resources.thorlabs.kinesis.structs),           |
| 447  | 570  |
| KNA_LowVoltageRange  | KNA_TTIARangeReading   |
| (class in msl.equipment.resources.thorlabs.kinesis.enums),           | (class in msl.equipment.resources.thorlabs.kinesis.structs),           |
| 447  | 570  |
| KNA_MMIParams  | KNA_TrigDisabled   |
| (class in msl.equipment.resources.thorlabs.kinesis.structs),         | (msl.equipment.resources.thorlabs.kinesis.enums.KNA_TrigDisabled       |

attribute), 449

KNA\_TriggerConfig (class in KPZ\_MMIParams (class in msl.equipment.resources.thorlabs.kinesis.structs), 571

KNA\_TriggerPortMode (class in KPZ\_TrigDisabled msl.equipment.resources.thorlabs.kinesis.enums), 449

KNA\_TriggerPortPolarity (class in KPZ\_TriggerConfig (class in msl.equipment.resources.thorlabs.kinesis.structs), 449

KNA\_TrigIn\_GPI KPZ\_TriggerPortMode (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

KNA\_TrigIn\_VoltageStepDown KPZ\_TriggerPortPolarity (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

KNA\_TrigIn\_VoltageStepUp KPZ\_TrigIn\_GPI (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

KNA\_TrigOut\_GPO KPZ\_TrigIn\_VoltageStepDown (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

KNA\_TrigPolarityHigh KPZ\_TrigIn\_VoltageStepUp (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 450

KNA\_TrigPolarityLow KPZ\_TrigOut\_GPO (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 450

KNA\_VoltageRange\_10v KPZ\_TrigPolarityHigh (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 447

KNA\_VoltageRange\_CH1\_150v KPZ\_TrigPolarityLow (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 448

KNA\_VoltageRange\_CH1\_75v KPZ\_WheelChangeRate (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 448

KNA\_VoltageRange\_CH2\_150v KPZ\_WheelDirectionSense (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 448

KNA\_VoltageRange\_CH2\_75v KPZ\_WheelMode (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 448

KNA\_WheelAdjustRate (class in KPZ\_WM\_High (msl.equipment.resources.thorlabs.kinesis.enums), 449

KNA\_WM\_High (msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

KNA\_WM\_Low (msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

KNA\_WM\_Medium (msl.equipment.resources.thorlabs.kinesis.enums), attribute), 449

attribute), 451  
 KPZ\_WM\_MoveAtVoltage (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KPZ_WM_MoveAtVoltage`), 451  
 KPZ\_WM\_Negative (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KPZ_WM_Negative`), 451  
 KPZ\_WM\_Positive (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KPZ_WM_Positive`), 451  
 KPZ\_WM\_SetVoltage (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KPZ_WM_SetVoltage`), 451  
 KSC\_MMIParams (class in `mssl.equipment.resources.thorlabs.kinesis.structs`), 576  
 KSC\_TriggerDisabled (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSC_TriggerDisabled`), 456  
 KSC\_TriggerConfig (class in `mssl.equipment.resources.thorlabs.kinesis.structs`), 576  
 KSC\_TriggerPortMode (class in `mssl.equipment.resources.thorlabs.kinesis.enums`), 456  
 KSC\_TriggerPortPolarity (class in `mssl.equipment.resources.thorlabs.kinesis.enums`), 456  
 KSC\_TrigIn\_GPI (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSC_TrigIn_GPI`), 456  
 KSC\_TrigOut\_GPO (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSC_TrigOut_GPO`), 456  
 KSC\_TrigPolarityHigh (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSC_TrigPolarityHigh`), 456  
 KSC\_TrigPolarityLow (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSC_TrigPolarityLow`), 456  
 KSG\_MMIParams (class in `mssl.equipment.resources.thorlabs.kinesis.structs`), 577  
 KSG\_TriggerDisabled (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TriggerDisabled`), 458  
 KSG\_TriggerConfig (class in `mssl.equipment.resources.thorlabs.kinesis.structs`), 577  
 KSG\_TriggerPortMode (class in `mssl.equipment.resources.thorlabs.kinesis.enums`), 458  
 KSG\_TriggerPortPolarity (class in `mssl.equipment.resources.thorlabs.kinesis.enums`), 458  
 KSG\_TrigIn\_GPI (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigIn_GPI`), 458  
 KSG\_TrigOut\_BetweenLimits (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigOut_BetweenLimits`), 458  
 KSG\_TrigOut\_GPO (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigOut_GPO`), 458  
 KSG\_TrigOut\_LessThanLowerLimit (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigOut_LessThanLowerLimit`), 458  
 KSG\_TrigOut\_LessThanUpperLimit (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigOut_LessThanUpperLimit`), 458  
 KSG\_TrigOut\_MoreThanLowerLimit (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigOut_MoreThanLowerLimit`), 458  
 KSG\_TrigOut\_MoreThanUpperLimit (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigOut_MoreThanUpperLimit`), 458  
 KSG\_TrigPolarityHigh (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigPolarityHigh`), 459  
 KSG\_TrigPolarityLow (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KSG_TrigPolarityLow`), 459  
 KST\_Stages (class in `mssl.equipment.resources.thorlabs.kinesis.enums.KST_Stages`), 456  
 LabJack\_050 (class in `mssl.equipment.resources.thorlabs.kinesis.motion_controller.LabJack_050`), 552  
 LabJack\_490 (class in `mssl.equipment.resources.thorlabs.kinesis.motion_controller.LabJack_490`), 552  
 Laser (class in `mssl.equipment.resources.thorlabs.kinesis.messages.Laser`), 550  
 last\_button\_press() (method in `mssl.equipment.resources.picotech.picoscope.ps2000.PicoScope`), 330  
 LAST\_USB (class in `mssl.equipment.resources.picotech.picoscope.ps2000.PicoScope`), 329

LAST\_USB (*msl.equipment.resources.picotech.picoscope.enums.PS2000* attribute), 332  
 lastNotUsed (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits* attribute), 560  
 lastNotUsed (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits* attribute), 560  
 lastNotUsed (*msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessPositionLoopParameters* attribute), 559  
 lastNotUsed (*msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessTrackSatelliteParameters* attribute), 559  
 lastNotUsed (*msl.equipment.resources.thorlabs.kinesis.structs.MOT\_VelocityProfilePublishingMessage* property), 557  
 latest\_calibration (*msl.equipment.record\_types.EquipmentRecord* property), 75  
 LD\_AnodeGrounded (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_Polarity* attribute), 444  
 LD\_CathodeGrounded (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_Polarity* attribute), 444  
 LD\_DisplayUnits (class in *msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits*), 443  
 LD\_ExternalSignal (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags* attribute), 443  
 LD\_ILD (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits* attribute), 443  
 LD\_ILim (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits* attribute), 443  
 LD\_InputSourceFlags (class in *msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags*), 442  
 LD\_IPD (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits* attribute), 443  
 LD\_PLD (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_DisplayUnits* attribute), 443  
 LD\_POLARITY (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 444  
 LD\_Potentiometer (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags* attribute), 443  
 LD\_SoftwareOnly (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_InputSourceFlags* attribute), 443  
 LD\_TIA\_100uA (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_TIA\_RANGES* attribute), 443  
 LD\_TIA\_10mA (*msl.equipment.resources.thorlabs.kinesis.enums.LD\_TIA\_RANGES* attribute), 444



LF (*msl.equipment.connection\_message\_based.ConnectionMessageBased* attribute), 23  
 load (*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.Connection* method), 534  
 LIMIT\_MODES (*msl.equipment.resources.mks\_instruments.pr4000b.PR4000B* attribute), 142  
 limitSwitch (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.BenchtopStepperMotor* attribute), 557  
 LinearRange (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors* attribute), 424  
 list\_resources() (in module *msl.equipment*), 17  
 LLimit (*msl.equipment.resources.nkt.nktpdll.ParameterSetType* attribute), 154  
 load\_calibration\_file() (*msl.equipment.resources.optronic\_laboratories.ol756ocx.Ol756ocx* method), 207  
 load\_named\_settings() (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.BenchtopStepperMotor* method), 401  
 load\_named\_settings() (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors* method), 477  
 load\_named\_settings() (*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.Connection* method), 500  
 load\_named\_settings() (*msl.equipment.resources.thorlabs.kinesis.kcube\_solenooid.Connection* method), 518  
 load\_named\_settings() (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor* method), 534  
 load\_sdk() (*msl.equipment.resources.nkt.nktpdll.NKT* static method), 163  
 load\_settings() (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.BenchtopStepperMotor* method), 401  
 load\_settings() (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors* method), 477  
 load\_settings() (*msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.Connection* method), 500  
 load\_settings() (*msl.equipment.resources.thorlabs.kinesis.kcube\_solenooid.Connection* method), 518  
 load\_settings() (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor* method), 518  
 local() (*msl.equipment.connection\_tcpip\_vxi11.ConnectionTCPIP* method), 207  
 LocalIp (*msl.equipment.resources.avantes.avaspec.Avantes.BroadcastAnswer* attribute), 93  
 lock() (*msl.equipment.connection\_tcpip\_hislip.ConnectionTCPIP* method), 41  
 lock() (*msl.equipment.connection\_tcpip\_vxi11.ConnectionTCPIP* method), 41  
 lock\_flipper\_filament (*msl.equipment.resources.mks\_instruments.pr4000b.PR4000B* method), 147  
 lock\_status() (*msl.equipment.connection\_tcpip\_hislip.ConnectionTCPIP* method), 38  
 lock\_timeout (*msl.equipment.connection\_tcpip\_hislip.ConnectionTCPIP* property), 38  
 lock\_timeout (*msl.equipment.connection\_tcpip\_vxi11.ConnectionTCPIP* property), 40  
 log\_critical() (*msl.equipment.connection.Connection* static method), 21  
 log\_debug() (*msl.equipment.connection.Connection* static method), 21  
 log\_errcheck() (*msl.equipment.connection.Connection* static method), 21  
 log\_error() (*msl.equipment.connection.Connection* static method), 21  
 log\_info() (*msl.equipment.connection.Connection* static method), 21  
 log\_warning() (*msl.equipment.connection.Connection* static method), 21  
 LONG\_PRESS (*msl.equipment.resources.picotech.picoscope.enums.PicoscopeEnums* attribute), 330  
 Long\_Travel\_Stage (*msl.equipment.resources.thorlabs.kinesis.motion\_control.motion\_control.MotionControl* attribute), 556  
 loopMode (*msl.equipment.resources.thorlabs.kinesis.structs.MOT\_Motor* attribute), 566  
 LOST\_DATA (*msl.equipment.resources.picotech.picoscope.ps2000.PicoscopePs2000* attribute), 330  
 LOST\_DATA (*msl.equipment.resources.picotech.picoscope.ps3000.PicoscopePs3000* attribute), 330

attribute), 333  
 LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A (class in msl.equipment.resources.thorlabs.kinesis.enums.LS\_Input attribute), 337  
 LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000A (class in msl.equipment.resources.thorlabs.kinesis.enums), attribute), 338  
 LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps5000.PicoScope5000 attribute), 340  
 LOST\_DATA (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000A attribute), 341  
 LOW (msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection attribute), 240  
 LOW (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalDirection attribute), 257  
 LowerLimit (msl.equipment.resources.thorlabs.kinesis.structs.KSGqTipgenConfig (class in msl.equipment.resources.thorlabs.kinesis.enums.LS\_Input attribute), 577  
 lowGearAcceleration (msl.equipment.resources.thorlabs.kinesis.structs.MQTI\_JoystickParameters attribute), 558  
 lowGearMaxVelocity (msl.equipment.resources.thorlabs.kinesis.structs.MQTI\_JoystickParameters attribute), 558  
 lowPassFilterCutOffFreq (msl.equipment.resources.thorlabs.kinesis.structs.QD\_LoopParameters attribute), 573  
 lowPassFilterCutOffFreq (msl.equipment.resources.thorlabs.kinesis.structs.QD\_LoopParameters attribute), 573  
 lowPassFilterEnabled (msl.equipment.resources.thorlabs.kinesis.structs.QD\_LoopParameters attribute), 573  
 lowPassFilterEnabled (msl.equipment.resources.thorlabs.kinesis.structs.QD\_LoopParameters attribute), 573  
 lowPassFilterEnabled (msl.equipment.resources.thorlabs.kinesis.structs.QD\_LoopParameters attribute), 573  
 lowVoltageOutputRoute (msl.equipment.resources.thorlabs.kinesis.structs.KNA\_IQ\_Equipment attribute), 571  
 lowVoltageOutputRoute (msl.equipment.resources.thorlabs.kinesis.structs.NT\_IQ\_Equipment attribute), 563  
 lowVoltageOutputRoute (msl.equipment.resources.thorlabs.kinesis.structs.NT\_IQ\_Equipment attribute), 574  
 lowVoltageOutRange (msl.equipment.resources.thorlabs.kinesis.structs.KNA\_IQ\_Equipment attribute), 571  
 lowVoltageOutRange (msl.equipment.resources.thorlabs.kinesis.structs.NT\_IQ\_Equipment attribute), 563  
 LS\_DisplayUnits (class in msl.equipment.resources.thorlabs.kinesis.enums), 460  
 LS\_ExternalSignal (class in msl.equipment.resources.thorlabs.kinesis.enums.LS\_Input attribute), 445  
 LS\_mAmps (msl.equipment.resources.thorlabs.kinesis.enums.LS\_Display attribute), 445  
 LS\_mDb (msl.equipment.resources.thorlabs.kinesis.enums.LS\_Display attribute), 445  
 LS\_mWatts (msl.equipment.resources.thorlabs.kinesis.enums.LS\_Display attribute), 445  
 LS\_Potentiometer (class in msl.equipment.resources.thorlabs.kinesis.enums.LS\_Input attribute), 445  
 LS\_SoftwareOnly (class in msl.equipment.resources.thorlabs.kinesis.enums.LS\_Input attribute), 445  
 LUTdiameter (msl.equipment.resources.thorlabs.kinesis.structs.NT\_IQ\_Equipment attribute), 564  
 LUTValueDelay (msl.equipment.resources.thorlabs.kinesis.structs.NT\_IQ\_Equipment attribute), 564  
 m\_aCalibConvers (msl.equipment.resources.avantes.avaspec.Avantes.SpectrumCalibration attribute), 106  
 m\_aCalibConvers (msl.equipment.resources.avantes.avaspec.SpectrumCalibration attribute), 95  
 m\_aFit (msl.equipment.resources.avantes.avaspec.Avantes.DetectorType attribute), 95  
 m\_aFit (msl.equipment.resources.avantes.avaspec.TecControlType attribute), 98  
 m\_aFit (msl.equipment.resources.avantes.avaspec.TecControlType attribute), 97  
 m\_aHighNLCounts (msl.equipment.resources.avantes.avaspec.Avantes.DetectorType attribute), 105  
 m\_aHighNLCounts (msl.equipment.resources.avantes.avaspec.DetectorType attribute), 95

|   |   |
|---|---|
| m_aLowNLCounts  | m_ConfigVersion   |
| (msl.equipment.resources.avantes.avaspec.DetectorType attribute), 95                          | (msl.equipment.resources.avantes.avaspec.Avantes.DeviceConfigType attribute), 109             |
| m_AnalogHigh (msl.equipment.resources.avantes.avaspec.ProcessControlType attribute), 108      | m_ConfigVersion (msl.equipment.resources.avantes.avaspec.DeviceConfigType attribute), 99      |
| m_AnalogHigh (msl.equipment.resources.avantes.avaspec.ProcessControlType attribute), 98       | m_Control (msl.equipment.resources.avantes.avaspec.Avantes.MeasConfigType attribute), 107     |
| m_AnalogLow (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 108  | m_Control (msl.equipment.resources.avantes.avaspec.MeasConfigType attribute), 98              |
| m_AnalogLow (msl.equipment.resources.avantes.avaspec.ProcessControlType attribute), 98        | m_CorDynDark (msl.equipment.resources.avantes.avaspec.Avantes.MeasConfigType attribute), 107  |
| m_aNLCorrect (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 105 | m_CorDynDark (msl.equipment.resources.avantes.avaspec.MeasConfigType attribute), 96           |
| m_aNLCorrect (msl.equipment.resources.avantes.avaspec.DetectorType attribute), 95             | m_data (msl.equipment.resources.avantes.avaspec.Avantes.OemDataType attribute), 110           |
| m_aReserved (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 110  | m_data (msl.equipment.resources.avantes.avaspec.OemDataType attribute), 99                    |
| m_aReserved (msl.equipment.resources.avantes.avaspec.DeviceConfigType attribute), 99          | m_Date (msl.equipment.resources.avantes.avaspec.Avantes.TimeStampType attribute), 107         |
| m_aSpectrumCorrect  | m_DefectivePixels   |
| (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 106              | (msl.equipment.resources.avantes.avaspec.DetectorType attribute), 97                          |
| m_aSpectrumCorrect  | m_DefectivePixels   |
| (msl.equipment.resources.avantes.avaspec.SpectrumCalibrationType attribute), 96               | (msl.equipment.resources.avantes.avaspec.Avantes.DetectorType attribute), 105                 |
| m_aTemperature  | m_DefectivePixels   |
| (msl.equipment.resources.avantes.avaspec.Avantes.DeviceConfigType attribute), 110             | (msl.equipment.resources.avantes.avaspec.DetectorType attribute), 94                          |
| m_aTemperature  | m_Detector (msl.equipment.resources.avantes.avaspec.Avantes.DeviceConfigType attribute), 109  |
| (msl.equipment.resources.avantes.avaspec.DeviceConfigType attribute), 99                      | m_Detector (msl.equipment.resources.avantes.avaspec.DeviceConfigType attribute), 99           |
| m_aUserFriendlyId   | m_DhcpEnabled (msl.equipment.resources.avantes.avaspec.EthernetConfigType attribute), 108     |
| (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 110              | m_DigitalHigh (msl.equipment.resources.avantes.avaspec.Avantes.MeasConfigType attribute), 108 |
| m_aUserFriendlyId   | m_DigitalHigh (msl.equipment.resources.avantes.avaspec.ProcessControlType attribute), 98      |
| (msl.equipment.resources.avantes.avaspec.DeviceConfigType attribute), 99                      | m_DigitalLow (msl.equipment.resources.avantes.avaspec.Avantes.MeasConfigType attribute), 108  |
| m_BitMatrix (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 109  | m_DigitalLow (msl.equipment.resources.avantes.avaspec.ProcessControlType attribute), 98       |
| m_BitMatrix (msl.equipment.resources.avantes.avaspec.HeatmapRespType attribute), 99           | m_DynamicStorage  |
| m_CalibrationType   | (msl.equipment.resources.avantes.avaspec.IrradianceType attribute), 95                        |
| (msl.equipment.resources.avantes.avaspec.AvantecDeviceConfigType attribute), 106              | (msl.equipment.resources.avantes.avaspec.Avantes.DeviceConfigType attribute), 109             |
| m_CalibrationType   | m_DynamicStorage  |
| (msl.equipment.resources.avantes.avaspec.IrradianceType attribute), 95                        | m_DynamicStorage (msl.equipment.resources.avantes.avaspec.DeviceConfigType attribute), 106    |
| m_CalInttime (msl.equipment.resources.avantes.avaspec.SpectrumCalibrationType attribute), 95  | m_Enable (msl.equipment.resources.avantes.avaspec.Avantes.Dark                                |

attribute), 105  
 m\_Enable (msl.equipment.resources.avantes.avaspec.Avantes.StandAloneType  
 attribute), 107  
 m\_Enable (msl.equipment.resources.avantes.avaspec.Avantes.IrradianceType  
 attribute), 108  
 m\_Enable (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 94  
 m\_Enable (msl.equipment.resources.avantes.avaspec.StandAloneType), 96  
 attribute), 97  
 m\_Enable (msl.equipment.resources.avantes.avaspec.TecControlType), 109  
 attribute), 97  
 m\_EthernetSettings  
 (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 109  
 m\_EthernetSettings  
 (msl.equipment.resources.avantes.avaspec.DeviceConfigType), 99  
 attribute), 99  
 m\_ExtOffset (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 105  
 m\_ExtOffset (msl.equipment.resources.avantes.avaspec.DetectorType), 94  
 attribute), 94  
 m\_FiberDiameter  
 (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 106  
 m\_FiberDiameter  
 (msl.equipment.resources.avantes.avaspec.IrradianceType), 94  
 attribute), 96  
 m\_ForgetPercentage  
 (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 105  
 m\_ForgetPercentage  
 (msl.equipment.resources.avantes.avaspec.DetectorType), 94  
 attribute), 94  
 m\_Gain (msl.equipment.resources.avantes.avaspec.Avantes.DetectorType), 99  
 attribute), 105  
 m\_Gain (msl.equipment.resources.avantes.avaspec.DetectorType), 109  
 attribute), 94  
 m\_Gateway (msl.equipment.resources.avantes.avaspec.Avantes.EthernetSettingsType  
 attribute), 109  
 m\_Gateway (msl.equipment.resources.avantes.avaspec.EthernetSettingsType), 97  
 attribute), 98  
 m\_IntegrationDelay  
 (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 107  
 m\_IntegrationDelay  
 (msl.equipment.resources.avantes.avaspec.MeasConfigType), 96  
 attribute), 96  
 m\_IntegrationTime  
 (msl.equipment.resources.avantes.avaspec.AvgIntensityCalibType  
 attribute), 107  
 m\_IntegrationTime  
 (msl.equipment.resources.avantes.avaspec.MeasConfigType), 96  
 attribute), 96  
 m\_IpAddr (msl.equipment.resources.avantes.avaspec.Avantes.EthernetSettingsType), 109  
 attribute), 98  
 m\_Irradiance (msl.equipment.resources.avantes.avaspec.DeviceConfigType), 99  
 m\_LaserDelay (msl.equipment.resources.avantes.avaspec.Avantes.IrradianceType), 105  
 m\_LaserDelay (msl.equipment.resources.avantes.avaspec.ControlSettingsType), 94  
 m\_LaserWaveLength  
 (msl.equipment.resources.avantes.avaspec.Avantes.ControlSettingsType), 94  
 m\_LaserWaveLength  
 (msl.equipment.resources.avantes.avaspec.ControlSettingsType), 94  
 m\_LaserWidth (msl.equipment.resources.avantes.avaspec.Avantes.ControlSettingsType), 104  
 m\_LaserWidth (msl.equipment.resources.avantes.avaspec.ControlSettingsType), 94  
 m\_Len (msl.equipment.resources.avantes.avaspec.Avantes.DeviceConfigType), 109  
 m\_Len (msl.equipment.resources.avantes.avaspec.DeviceConfigType), 99  
 m\_LinkStatus (msl.equipment.resources.avantes.avaspec.Avantes.EthernetSettingsType), 109  
 m\_LinkStatus (msl.equipment.resources.avantes.avaspec.EthernetSettingsType), 97  
 m\_Meas (msl.equipment.resources.avantes.avaspec.Avantes.StandAloneType), 96  
 m\_Meas (msl.equipment.resources.avantes.avaspec.StandAloneType), 97  
 m\_Mode (msl.equipment.resources.avantes.avaspec.TriggerType), 106  
 m\_Mode (msl.equipment.resources.avantes.avaspec.TriggerType), 96  
 m\_NetMask (msl.equipment.resources.avantes.avaspec.Avantes.EthernetSettingsType), 109  
 m\_NetMask (msl.equipment.resources.avantes.avaspec.EthernetSettingsType), 98  
 m\_NLEnable (msl.equipment.resources.avantes.avaspec.Avantes.DeviceConfigType), 96



attribute), 105  
 m\_NLEnable (msl.equipment.resources.avantes.avaspec.SaturationType  
 attribute), 94  
 m\_Nmsr (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 108  
 m\_Nmsr (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 107  
 m\_Nmsr (msl.equipment.resources.avantes.avaspec.DynamicStorageType  
 attribute), 97  
 m\_Nmsr (msl.equipment.resources.avantes.avaspec.StandAloneType  
 attribute), 97  
 m\_NrAverages (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 107  
 m\_NrAverages (msl.equipment.resources.avantes.avaspec.MeasConfigType  
 attribute), 96  
 m\_NrPixels (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 105  
 m\_NrPixels (msl.equipment.resources.avantes.avaspec.DetectionType), 96  
 m\_OemData (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 110  
 m\_OemData (msl.equipment.resources.avantes.avaspec.DeviceConfigType  
 attribute), 99  
 m\_Offset (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 105  
 m\_Offset (msl.equipment.resources.avantes.avaspec.DetectionType), 95  
 m\_ProcessControl  
 (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 110  
 m\_ProcessControl  
 (msl.equipment.resources.avantes.avaspec.DeviceConfigType  
 attribute), 99  
 m\_Reflectance (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 110  
 m\_Reflectance (msl.equipment.resources.avantes.avaspec.DeviceConfigType  
 attribute), 99  
 m\_Reserved (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 105  
 m\_Reserved (msl.equipment.resources.avantes.avaspec.SpectrumCorrectType  
 attribute), 108  
 m\_Reserved (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 109  
 m\_Reserved (msl.equipment.resources.avantes.avaspec.DetectionType), 95  
 m\_Reserved (msl.equipment.resources.avantes.avaspec.DynamicStorageType  
 attribute), 97  
 m\_Reserved (msl.equipment.resources.avantes.avaspec.HearthRespType  
 attribute), 99  
 m\_SaturationDetection  
 (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 107  
 m\_SaturationDetection  
 (msl.equipment.resources.avantes.avaspec.MeasConfigType  
 attribute), 96  
 m\_SensorType (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 105  
 m\_SensorType (msl.equipment.resources.avantes.avaspec.DetectionType), 95  
 m\_Setpoint (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 108  
 m\_Setpoint (msl.equipment.resources.avantes.avaspec.TecControlType  
 attribute), 98  
 m\_Smoothing (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 107  
 m\_Smoothing (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 96  
 m\_Smoothing (msl.equipment.resources.avantes.avaspec.SpectrumCorrectType  
 attribute), 105  
 m\_SmoothModel (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 105  
 m\_SmoothModel (msl.equipment.resources.avantes.avaspec.SpectrumCorrectType  
 attribute), 95  
 m\_SmoothPix (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 106  
 m\_SmoothPix (msl.equipment.resources.avantes.avaspec.SpectrumCorrectType  
 attribute), 95  
 m\_Source (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 106  
 m\_Source (msl.equipment.resources.avantes.avaspec.TriggerType  
 attribute), 96  
 m\_SourceType (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 106  
 m\_SourceType (msl.equipment.resources.avantes.avaspec.TriggerType  
 attribute), 96  
 m\_SpectrumCorrect  
 (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 110  
 m\_SpectrumCorrect  
 (msl.equipment.resources.avantes.avaspec.DeviceConfigType  
 attribute), 99  
 m\_StandAlone (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 110  
 m\_StandAlone (msl.equipment.resources.avantes.avaspec.DeviceConfigType  
 attribute), 97  
 m\_StartPixel (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 107  
 m\_StartPixel (msl.equipment.resources.avantes.avaspec.MeasConfigType  
 attribute), 96  
 m\_StopPixel (msl.equipment.resources.avantes.avaspec.Avantec.MeasConfigType  
 attribute), 107

662 Index

MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000DigitalPort attribute), 232

MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000ElementId attribute), 227

MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000SetupType attribute), 228

MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000TimeUnit attribute), 226

MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000TriggerDirection attribute), 227

MAX (msl.equipment.resources.picotech.picoscope.enums.PS2000TriggerState attribute), 230

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000AChannel, 246BufferIndex attribute), 248

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort attribute), 257

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ADigitalPort attribute), 249

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000AElementId attribute), 252

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000AHoldOffType attribute), 259

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000AIndexMode attribute), 255

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ASetupType attribute), 253

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ATimeUnit attribute), 253

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ATriggerState attribute), 257

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType attribute), 254

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000ElementId attribute), 246

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000TimeUnit attribute), 244

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerDirection attribute), 245

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000TriggerState attribute), 247

MAX (msl.equipment.resources.picotech.picoscope.enums.PS3000WaveType attribute), 243

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000AChannel, 288BufferIndex attribute), 272

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000AElementId attribute), 273

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndexMode attribute), 277

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ASetupType attribute), 274

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ATimeUnit attribute), 273

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000DigitalPort attribute), 274

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000ElementId attribute), 279

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000SetupType attribute), 275

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000TimeUnit attribute), 259

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000TriggerDirection attribute), 263

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000TriggerState attribute), 266

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSAChannel, 246BufferIndex attribute), 248

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSADigitalPort attribute), 257

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSAElementId attribute), 267

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSAHoldOffType attribute), 267

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSAIndexMode attribute), 267

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSASetupType attribute), 269

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSATimeUnit attribute), 269

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSATriggerState attribute), 269

MAX (msl.equipment.resources.picotech.picoscope.enums.PS4000PSAWaveType attribute), 269

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AChannel, 288BufferIndex attribute), 288

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AElementId attribute), 287

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndexMode attribute), 287

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ASetupType attribute), 289

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ATimeUnit attribute), 289

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ATriggerDirection attribute), 289

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000ATriggerState attribute), 289

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000IndexMode attribute), 289

MAX (msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType attribute), 289

MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000Channel attribute), 289

|  |  |
|--|--|
| MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000a.PicoScope attribute), 304                          | MAX_ANALOGUE_OFFSET_5V_20V (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute), 335 |
| MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000a.PicoScope attribute), 308                          | MAX_ANALOGUE_OFFSET_5V_20V (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute), 339 |
| MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000a.PicoScope attribute), 305                          | MAX_ANALOGUE_OFFSET_5V_20V (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute), 342 |
| MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000a.PicoScope attribute), 309                          | MAX_ANALOGUE_OFFSET_5V_20V (msl.equipment.resources.picotech.picoscope.ps6000.PicoScope attribute), 344  |
| MAX (msl.equipment.resources.picotech.picoscope.enums.PS6000a.PicoScope attribute), 306                          | max_channel_count() (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope attribute), 270  |
| MAX_4_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope attribute), 261               | MAX_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope attribute), 225         |
| MAX_ANALOGUE_OFFSET_500MV_2V (msl.equipment.resources.picotech.picoscope.enums.PS2000a.PicoScope attribute), 331 | MAX_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS2000a.PicoScope attribute), 249         |
| MAX_ANALOGUE_OFFSET_500MV_2V (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute), 335       | MAX_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS3000a.PicoScope attribute), 271         |
| MAX_ANALOGUE_OFFSET_500MV_2V (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute), 339       | MAX_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope attribute), 260         |
| MAX_ANALOGUE_OFFSET_500MV_2V (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute), 342       | MAX_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS5000a.PicoScope attribute), 285         |
| MAX_ANALOGUE_OFFSET_500MV_2V (msl.equipment.resources.picotech.picoscope.ps6000.PicoScope attribute), 343        | MAX_CHANNELS (msl.equipment.resources.picotech.picoscope.enums.PS6000.PicoScope attribute), 302          |
| MAX_ANALOGUE_OFFSET_50MV_200MV (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute), 331     | MAX_CLIENTS (msl.equipment.hislip.ErrorType attribute), 52   |
| MAX_ANALOGUE_OFFSET_50MV_200MV (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute), 335     | MAX_DATA_TYPES (msl.equipment.resources.picotech.pt104.Pt104DataType attribute), 33                      |
| MAX_ANALOGUE_OFFSET_50MV_200MV (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute), 339     | MAX_DELAY_COUNT (msl.equipment.resources.picotech.picoscope.ps4000.PicoScope attribute), 240             |
| MAX_ANALOGUE_OFFSET_50MV_200MV (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute), 342     | MAX_DELAY_COUNT (msl.equipment.resources.picotech.picoscope.ps6000.PicoScope attribute), 260             |
| MAX_ANALOGUE_OFFSET_50MV_200MV (msl.equipment.resources.picotech.picoscope.ps6000.PicoScope attribute), 343      | MAX_DELAY_COUNT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute), 240            |
| MAX_ANALOGUE_OFFSET_5V_20V (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute), 331         | MAX_ETS_CYCLES (msl.equipment.hislip.ErrorType attribute), 52  |



|   |   |
|---|---|
| <code>(msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000</code><br>attribute), 344  | <code>MAX_PULSE_WIDTH_QUALIFIER_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000</code><br>attribute), 333   |
| <code>MAX_ETS_CYCLES_INTERLEAVE_RATIO</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000a</code><br>attribute), 330 | <code>MAX_PULSE_WIDTH_QUALIFIER_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000</code><br>attribute), 337   |
| <code>MAX_ETS_CYCLES_INTERLEAVE_RATIO</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000a</code><br>attribute), 333 | <code>MAX_PULSE_WIDTH_QUALIFIER_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000a</code><br>attribute), 339 |
| <code>MAX_EXTRA_RESISTANCES</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000</code><br>attribute), 261             | <code>MAX_PULSE_WIDTH_QUALIFIER_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps5000.PicoScope5000</code><br>attribute), 340   |
| <code>MAX_HOLDOFF_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000a</code><br>attribute), 333               | <code>MAX_PULSE_WIDTH_QUALIFIER_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope5000a</code><br>attribute), 341 |
| <code>MAX_INTERLEAVE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps6000.PicoScope6000</code><br>attribute), 344                    | <code>MAX_PULSE_WIDTH_QUALIFIER_COUNT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps6000.PicoScope6000</code><br>attribute), 343   |
| <code>MAX_LOGIC_LEVEL</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000a</code><br>attribute), 331                 | <code>MAX_RANGES</code> ( <code>msl.equipment.resources.picotech.picoscope.enums.HislipConnectionType</code> attribute), 260                        |
| <code>MAX_LOGIC_LEVEL</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000</code><br>attribute), 334                   | <code>max_read_size</code> ( <code>msl.equipment.connection_message_based.ConnectionMessageBased</code> attribute), 23                              |
| <code>MAX_NR_PIXELS</code> ( <code>msl.equipment.resources.avantes.avaspec.AvaSpec</code> attribute), 100   | <code>max_read_size</code> ( <code>msl.equipment.connection_prologix.ConnectionPrologix</code> attribute), 28                                       |
| <code>MAX_OVERSAMPLE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000</code><br>attribute), 329                    | <code>max_read_size</code> ( <code>msl.equipment.connection_tcpip_hislip.ConnectionTcpipHislip</code> attribute), 37                                |
| <code>MAX_OVERSAMPLE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000</code><br>attribute), 333                    | <code>max_read_size</code> ( <code>msl.equipment.connection_zeromq.ConnectionZeromq</code> attribute), 43   |
| <code>MAX_OVERSAMPLE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000</code><br>attribute), 333                    | <code>MAX_RESISTANCES</code><br>( <code>msl.equipment.resources.picotech.picoscope.enums.PS4000aResistances</code> attribute), 261                  |
| <code>MAX_OVERSAMPLE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope3000</code><br>attribute), 334                    | <code>MAX_SIG_GEN_BUFFER_SIZE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000a</code> attribute), 331            |
| <code>MAX_OVERSAMPLE_12BIT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000</code><br>attribute), 336              | <code>MAX_SIG_GEN_BUFFER_SIZE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000a</code> attribute), 335            |
| <code>MAX_OVERSAMPLE_8BIT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000</code><br>attribute), 336               | <code>MAX_SIG_GEN_BUFFER_SIZE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000</code> attribute), 337              |
| <code>MAX_OVERSAMPLE_8BIT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps5000.PicoScope5000</code><br>attribute), 340               | <code>MAX_SIG_GEN_BUFFER_SIZE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000a</code> attribute), 339            |
| <code>MAX_OVERSAMPLE_8BIT</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps6000.PicoScope6000</code><br>attribute), 343               | <code>MAX_SIG_GEN_BUFFER_SIZE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps5000.PicoScope5000</code> attribute), 340              |
| <code>MAX_PROBES</code> ( <code>msl.equipment.resources.picotech.picoscope.ps6000.PicoScope6000</code> attribute), 262                              | <code>MAX_SIG_GEN_BUFFER_SIZE</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps6000.PicoScope6000</code> attribute), 343              |
| <code>MAX_PULSE_WIDTH</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.filMAX_STEP_GEN_FREQ</code> attribute), 463                        | <code>MAX_STEP_GEN_FREQ</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope2000a</code> attribute), 343                  |

|  |  |
|--|--|
| <code>attribute</code> ), 331  | <code>MAX_TRIGGER_SOURCES</code>   |
| <code>MAX_SIG_GEN_FREQ</code>  | <code>(msl.equipment.resources.picotech.picoscope.enums.PS3000a.PicoScope23000A</code>                   |
| <code>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope23000A</code>                         | <code>attribute</code> ), 334  |
| <code>MAX_SIG_GEN_FREQ</code>  | <code>(msl.equipment.resources.picotech.picoscope.enums.PS3000a.PicoScope2400</code>                     |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope2400</code>                           | <code>attribute</code> ), 337  |
| <code>MAX_SIG_GEN_FREQ_4262</code>   | <code>(msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope24000</code>                    |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope24000</code>                          | <code>attribute</code> ), 337  |
| <code>MAX_SIGGEN_FREQ</code>   | <code>(msl.equipment.resources.picotech.picoscope.enums.PS4000a.PicoScope23000</code>                    |
| <code>(msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope23000</code>                          | <code>attribute</code> ), 330  |
| <code>MAX_SIGGEN_FREQ</code>   | <code>(msl.equipment.resources.picotech.picoscope.enums.PS5000a.PicoScope23000</code>                    |
| <code>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope23000</code>                          | <code>attribute</code> ), 333  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>(msl.equipment.resources.picotech.picoscope.enums.PS5000a.PicoScope282000A</code>                  |
| <code>(msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope282000A</code>                        | <code>attribute</code> ), 331  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>(msl.equipment.resources.picotech.picoscope.enums.PS6000a.PicoScope263000A</code>                  |
| <code>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope263000A</code>                        | <code>attribute</code> ), 335  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_UNITS</code> ( <code>msl.equipment.resources.picotech.picoscope.ps2000.PicoScope24000</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope24000</code>                          | <code>attribute</code> ), 329  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_UNITS</code> ( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope24000</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope24000</code>                          | <code>attribute</code> ), 332  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_UNITS_OPENED</code>  |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope24000</code>                          | <code>resources.picotech.errors.PS2000Error</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope24000</code>                          | <code>attribute</code> ), 219  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_UNITS_OPENED</code>  |
| <code>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope25000</code>                          | <code>resources.picotech.errors.PS3000Error</code>   |
| <code>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope25000</code>                          | <code>attribute</code> ), 220  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_VALUE</code> ( <code>msl.equipment.resources.picotech.picoscope.ps2000.PicoScope235000A</code> |
| <code>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope235000A</code>                        | <code>attribute</code> ), 342  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_VALUE</code> ( <code>msl.equipment.resources.picotech.picoscope.ps3000.PicoScope260000</code>  |
| <code>(msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope260000</code>                         | <code>attribute</code> ), 333  |
| <code>MAX_SWEEPS_SHOTS</code>  | <code>MAX_VALUE</code> ( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope260000</code>  |
| <code>(msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope260000</code>                         | <code>attribute</code> ), 336  |
| <code>MAX_TEMP_SENSORS</code>  | <code>MAX_VALUE</code> ( <code>msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope260000</code> |
| <code>(msl.equipment.resources.avantes.avaspec.Avantes</code>  | <code>attribute</code> ), 338  |
| <code>attribute</code> ), 100  | <code>MAX_VALUE</code> ( <code>msl.equipment.resources.picotech.picoscope.ps5000.PicoScope260000</code>  |
| <code>MAX_TEMPERATURES</code>  | <code>attribute</code> ), 340  |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope260000</code>                         | <code>MAX_UNITS</code> ( <code>msl.equipment.resources.picotech.picoscope.ps6000.PicoScope260000</code>  |
| <code>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope260000</code>                         | <code>attribute</code> ), 343  |
| <code>MAX_TIMEBASE</code> ( <code>msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000</code> | <code>attribute</code> ), 329  |
| <code>MAX_TRANSIT_TIME</code>  | <code>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope260000</code>                         |
| <code>(msl.equipment.resources.thorlabs.kinesis.filmax.flipper.FBFTFlipper</code>                        | <code>attribute</code> ), 341  |
| <code>attribute</code> ), 463  | <code>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope260000</code>                         |
| <code>MAX_TRIGGER_SOURCES</code>   | <code>attribute</code> ), 341  |
| <code>(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope260000</code>                          | <code>MAX_WIDES2000CHANNELS</code>   |
| <code>(msl.equipment.resources.avantes.avaspec.Avantes</code>  | <code>attribute</code> ), 232  |

[attribute](#)), [100](#)  
**MAX\_WAVEFORMS\_PER\_SECOND** ([msl.equipment.resources.picotech.picoscope.enums.PS2000Error](#)), [100](#)  
[attribute](#)), [343](#)  
**MAX\_WIRES** ([msl.equipment.resources.picotech.pt104MEM\\_FAIL](#) ([msl.equipment.resources.picotech.errors.PS2000Error](#) [attribute](#)), [219](#)  
**maxAcceleration** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [557](#)  
**maxChannels** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [555](#)  
**maxDeceleration** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [557](#)  
**maxDiameter** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [562](#)  
**maximum\_message\_size** ([msl.equipment.hislip.AsyncMaximumMessageSizeResponse](#) [property](#)), [60](#)  
**maximum\_server\_message\_size** ([msl.equipment.hislip.HiSLIPClient](#) [property](#)), [67](#)  
**maximum\_value()** ([msl.equipment.resources.picotech.picoscope.picoscope.picoscope\\_a](#) [method](#)), [313](#)  
**maxPosition** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [557](#)  
**maxTrackingError** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [559](#)  
**maxVelocity** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [557](#)  
**maxVelocity** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [556](#)  
**maxXdemand** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [574](#)  
**maxYdemand** ([msl.equipment.resources.thorlabs.kinesis.structs.MOTIFilterStageAxisParameters](#) [attribute](#)), [574](#)  
**MeasConfigType** (class in [msl.equipment.resources.avantes.avaspec](#)), [96](#)  
**MeasurandRecord** (class in [msl.equipment.record\\_types](#)), [77](#)  
**measurands** ([msl.equipment.record\\_types.CalibrationRecord](#) [attribute](#)), [76](#)  
**measure()** ([msl.equipment.resources.avantes.avaspec](#) [method](#)), [116](#)  
**measure\_callback()** ([msl.equipment.resources.avantes.avaspec](#) [method](#)), [116](#)  
**MeasureCallback** (in module [msl.equipment.resources.thorlabs.kinesis.messages](#)), [550](#)

MessageType (*class in msl.equipment.hislip*), 50  
 MessageType (*class in msl.equipment.vxi11*), 586  
 MessageTypes (in module (msl.equipment.resources.picotech.picoscope.ps6000.PicoScope)), 344  
 (msl.equipment.resources.thorlabs.kinesis.messages), attribute), 344  
 549  
 MIN\_DWELL\_COUNT  
 MILLI\_TO\_MICRO (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 331  
 (msl.equipment.resources.avantes.avaspec.Avantes), attribute), 101  
 MIN\_DWELL\_COUNT  
 Min (msl.equipment.resources.nkt.nktpdll.DateTimeType (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 153  
 attribute), 335  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V MIN\_DWELL\_COUNT  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 331  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope), attribute), 337  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V MIN\_DWELL\_COUNT  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 335  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope), attribute), 339  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V MIN\_DWELL\_COUNT  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope), attribute), 339  
 (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 341  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V MIN\_DWELL\_COUNT  
 (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 342  
 (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 342  
 MIN\_ANALOGUE\_OFFSET\_500MV\_2V MIN\_DWELL\_COUNT  
 (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope), attribute), 344  
 (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope), attribute), 343  
 MIN\_ANALOGUE\_OFFSET\_50MV\_200MV MIN\_ETS\_CYCLES\_INTERLEAVE\_RATIO  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 331  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 330  
 MIN\_ANALOGUE\_OFFSET\_50MV\_200MV MIN\_ETS\_CYCLES\_INTERLEAVE\_RATIO  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 335  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 333  
 MIN\_ANALOGUE\_OFFSET\_50MV\_200MV MIN\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 339  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope), attribute), 339  
 MIN\_ANALOGUE\_OFFSET\_50MV\_200MV MIN\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 335  
 (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 339  
 MIN\_ANALOGUE\_OFFSET\_50MV\_200MV MIN\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 343  
 (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope), attribute), 343  
 MIN\_ANALOGUE\_OFFSET\_50MV\_200MV MIN\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 342  
 (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope), attribute), 344  
 MIN\_ANALOGUE\_OFFSET\_5V\_20V MIN\_ETS\_CYCLES\_INTERLEAVE\_RATIO  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 331  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 344  
 MIN\_ANALOGUE\_OFFSET\_5V\_20V MIN\_ILX\_INTTIME  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 335  
 (msl.equipment.resources.avantes.avaspec.Avantes), attribute), 101  
 MIN\_ANALOGUE\_OFFSET\_5V\_20V MIN\_LOGIC\_LEVEL  
 (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope), attribute), 339  
 (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope), attribute), 331  
 MIN\_ANALOGUE\_OFFSET\_5V\_20V MIN\_LOGIC\_LEVEL  
 (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope), attribute), 343  
 (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope), attribute), 343



|  |   |
|--|---|
| <a href="#">attribute</a> ), 334   | <a href="#">MIN_VALUE (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute)</a> , 340                |
| <a href="#">MIN_PULSE_WIDTH (msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper attribute)</a> , 463  | <a href="#">MIN_VALUE_FILTER_FLIPPER (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope attribute)</a> , 343 |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute)</a> , 331   | <a href="#">MIN_VALUE_16BIT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute)</a> , 341          |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute)</a> , 335   | <a href="#">MIN_VALUE_8BIT (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute)</a> , 341           |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute)</a> , 337   | <a href="#">MIN_WIRES (msl.equipment.resources.picotech.pt104.PT104 attribute)</a> , 340                                |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute)</a> , 339   | <a href="#">minDiameter (msl.equipment.resources.thorlabs.kinesis.structs.NT_CircleFit attribute)</a> , 562             |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute)</a> , 340   | <a href="#">minIn4000a.PicoScope (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope method)</a> , 313      |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps5000a.PicoScope attribute)</a> , 342   | <a href="#">min5000a.PicoScope (msl.equipment.resources.thorlabs.kinesis.structs.MOT_JogP attribute)</a> , 557          |
| <a href="#">MIN_SIG_GEN_BUFFER_SIZE (msl.equipment.resources.picotech.picoscope.ps6000a.PicoScope attribute)</a> , 343   | <a href="#">minVelocity (msl.equipment.resources.thorlabs.kinesis.structs.MOT_Veloc attribute)</a> , 5500A              |
| <a href="#">MIN_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute)</a> , 331          | <a href="#">minXdemand (msl.equipment.resources.thorlabs.kinesis.structs.QD attribute)</a> , 574                        |
| <a href="#">MIN_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute)</a> , 334          | <a href="#">minYdemand (msl.equipment.resources.thorlabs.kinesis.structs.QD attribute)</a> , 574                        |
| <a href="#">MIN_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute)</a> , 337          | <a href="#">MKSInstrumentsError</a> , 49  |
| <a href="#">MIN_SIGGEN_FREQ (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute)</a> , 330           | <a href="#">mode (msl.equipment.resources.thorlabs.kinesis.structs.KNA_TIAR attribute)</a> , 570                        |
| <a href="#">MIN_SIGGEN_FREQ (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute)</a> , 334           | <a href="#">mode (msl.equipment.resources.thorlabs.kinesis.structs.MOT_JogP attribute)</a> , 53000A                     |
| <a href="#">MIN_SIG_GEN_FREQ (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute)</a> , 337          | <a href="#">mode (msl.equipment.resources.thorlabs.kinesis.structs.MOT_Veloc attribute)</a> , 557                       |
| <a href="#">MIN_SIGGEN_FREQ (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute)</a> , 330           | <a href="#">mode4000a.PicoScope (msl.equipment.resources.thorlabs.kinesis.structs.NT_CircleF attribute)</a> , 562       |
| <a href="#">MIN_SIGGEN_FREQ (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute)</a> , 333           | <a href="#">mode (msl.equipment.resources.thorlabs.kinesis.structs.NT_TIARan attribute)</a> , 5200                      |
| <a href="#">MIN_TRANSIT_TIME (msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper attribute)</a> , 463 | <a href="#">mode (msl.equipment.resources.thorlabs.kinesis.structs.PZ_LUTWa attribute)</a> , 564                        |
| <a href="#">MIN_VALUE (msl.equipment.resources.picotech.picoscope.ps2000a.PicoScope attribute)</a> , 330                 | <a href="#">mode3000a.PicoScope (msl.equipment.resources.thorlabs.kinesis.structs.KLD_Trig attribute)</a> , 569         |
| <a href="#">MIN_VALUE (msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope attribute)</a> , 333                 | <a href="#">mode1 (msl.equipment.resources.thorlabs.kinesis.structs.KLS_Trig attribute)</a> , 570                       |
| <a href="#">MIN_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute)</a> , 336                 | <a href="#">mode2 (msl.equipment.resources.thorlabs.kinesis.structs.KLD_Trig attribute)</a> , 5692000                   |
| <a href="#">MIN_VALUE (msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope attribute)</a> , 338                 | <a href="#">mode2 (msl.equipment.resources.thorlabs.kinesis.structs.KLS_Trig attribute)</a> , 5703000                   |
|  | <a href="#">model (msl.equipment.record_types.ConnectionRecord attribute)</a> , 339                                     |
|  | <a href="#">model (msl.equipment.record_types.EquipmentRecord attribute)</a> , 339                                      |
|  | <a href="#">modelNumber (msl.equipment.resources.thorlabs.kinesis.structs.TL attribute)</a> , 5692000                   |

attribute), 556  
 modificationState  
     (msl.equipment.resources.thorlabs.kinesis.structs.TLIHardwareInformation  
     attribute), 556  
 Modular\_NanoTrak  
     (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl  
     attribute), 552  
 Modular\_Piezo (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl  
     attribute), 552  
 Modular\_Stepper\_Motor  
     (msl.equipment.resources.thorlabs.kinesis.motion\_control.MotionControl  
     attribute), 552  
 module  
     msl.equipment, 17  
     msl.equipment.config, 17  
     msl.equipment.connection, 20  
     msl.equipment.connection\_demo, 22  
     msl.equipment.connection\_message\_based, 22  
     msl.equipment.connection\_nidaq, 25  
     msl.equipment.connection\_prologix, 27  
     msl.equipment.connection\_pyvisa, 31  
     msl.equipment.connection\_sdk, 32  
     msl.equipment.connection\_serial, 33  
     msl.equipment.connection\_socket, 35  
     msl.equipment.connection\_tcpip\_hislip, 36  
     msl.equipment.connection\_tcpip\_vxi11, 39  
     msl.equipment.connection\_zeromq, 42  
     msl.equipment.constants, 43  
     msl.equipment.database, 45  
     msl.equipment.dns\_service\_discovery, 47  
     msl.equipment.exceptions, 48  
     msl.equipment.factory, 50  
     msl.equipment.hislip, 50  
     msl.equipment.resources, 81  
     msl.equipment.resources.aim\_tti, 84  
     msl.equipment.resources.aim\_tti.mx\_series, 84  
     msl.equipment.resources.avantes, 91  
     msl.equipment.resources.avantes.avaspec, 91  
     msl.equipment.resources.bentham, 121  
     msl.equipment.resources.bentham.benhw32, 122  
     msl.equipment.resources.bentham.benhw64, 124  
     msl.equipment.resources.bentham.errors, 125  
     msl.equipment.resources.bentham.tokens, 125  
     msl.equipment.resources.cmi, 125  
     msl.equipment.resources.cmi.sia3, 125  
     msl.equipment.resources.dataray, 127  
     msl.equipment.resources.dataray.datarayocx\_32, 127  
     msl.equipment.resources.dataray.datarayocx\_64, 127  
     msl.equipment.resources.dmm, 81  
     msl.equipment.resources.electron\_dynamics, 130  
     msl.equipment.resources.electron\_dynamics.tc\_serial, 130  
     msl.equipment.resources.energetiq, 135  
     msl.equipment.resources.energetiq.eq99, 135  
     msl.equipment.resources.mks\_instruments, 141  
     msl.equipment.resources.mks\_instruments.pr4000b, 141  
     msl.equipment.resources.nkt, 153  
     msl.equipment.resources.nkt.nktpdll, 153  
     msl.equipment.resources.omega, 186  
     msl.equipment.resources.omega.ithx, 186  
     msl.equipment.resources.optosigma, 190  
     msl.equipment.resources.optosigma.shot702, 190  
     msl.equipment.resources.optronic\_laboratories, 197  
     msl.equipment.resources.optronic\_laboratories.olb, 197  
     msl.equipment.resources.optronic\_laboratories.olc, 215  
     msl.equipment.resources.optronic\_laboratories.olc2, 216  
     msl.equipment.resources.picotech, 219  
     msl.equipment.resources.picotech.errors, 219  
     msl.equipment.resources.picotech.picoscope, 220  
     msl.equipment.resources.picotech.picoscope.callb, 220  
     msl.equipment.resources.picotech.picoscope.chann, 220

222  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.callbacks, 389  
 224  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.enums, 418  
 310  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.errors, 418  
 310  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.filter\_ 463  
 312  
 msl.equipment.resources.picotech.picoscope.mslequipment2k2k, 463  
 316  
 msl.equipment.resources.picotech.picoscope.mslequipment2k2k.resources.thorlabs.kinesis.integra 468  
 318  
 msl.equipment.resources.picotech.picoscope.mslequipmenttap, 490  
 329  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.kcube\_s 514  
 331  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.kcube\_s 523  
 332  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.message 549  
 334  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.motion\_ 551  
 336  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.structs 555  
 338  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.utils, 82  
 340  
 msl.equipment.resources.picotech.picoscope.mslequipment.vxi11, 586  
 340  
 monitorOPBandwidth  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.structs.PPC\_IC 550  
 341  
 attribute), 565  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.structs.P 350  
 343  
 attribute), 565  
 msl.equipment.resources.picotech.picoscope.mslequipment.resources.thorlabs.kinesis.structs.P 350  
 345  
 some() (msl.equipment.resources.princeton\_instruments.arc\_instr 362  
 msl.equipment.resources.picotech.pt104, method), 362  
 357  
 mono\_grating\_calc\_gadjust()  
 msl.equipment.resources.princeton\_instruments.mslequipment.resources.princeton\_instruments.arc\_instr 362  
 361  
 method), 362  
 msl.equipment.resources.princeton\_instruments.mslequipment.resources.princeton\_instruments.arc\_instr 362  
 361  
 (msl.equipment.resources.princeton\_instruments.arc\_instr 362  
 msl.equipment.resources.raicol, 388  
 msl.equipment.resources.raicol.raicol.mslequipment.resources.princeton\_instruments.arc\_instr 362  
 388  
 method), 362  
 msl.equipment.resources.thorlabs, 389  
 389  
 mono\_grating\_uninstall()  
 msl.equipment.resources.thorlabs.fwx2c, (msl.equipment.resources.princeton\_instruments.arc\_instr 362  
 578  
 method), 363  
 msl.equipment.resources.thorlabs.kinesis.mslequipment.resources.princeton\_instruments.arc\_instr 362  
 389  
 mono\_move\_steps()  
 msl.equipment.resources.thorlabs.kinesis.mslequipment.resources.princeton\_instruments.arc\_instr 362  
 389  
 mono\_reset() (msl.equipment.resources.princeton\_instruments.arc\_instr 362  
 msl.equipment.resources.thorlabs.kinesis.benchmark.mslequipment.resources.princeton\_instruments.arc\_instr 362  
 389  
 stepper\_motor,

`mono_restore_factory_settings()` (`msl.equipment.resources.thorlabs.kinesis.structs`),  
(`msl.equipment.resources.princeton_instruments.arc_566` `msl.instrument.PrincetonInstruments`  
`method`), 363 `MOT_Continuous`

`mono_scan_done()` (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_Jo`  
(`msl.equipment.resources.princeton_instruments.arc_423` `msl.instrument.PrincetonInstruments`  
`method`), 363 `MOT_CurrentLoopPhases` (class in  
`msl.equipment.resources.thorlabs.kinesis.enums`),

`mono_slit_home()` (`msl.equipment.resources.princeton_instruments.arc_423` `msl.instrument.PrincetonInstruments`  
`method`), 363 `MOT_CustomMotor`

`mono_slit_name()` (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_M`  
(`msl.equipment.resources.princeton_instruments.arc_423` `msl.instrument.PrincetonInstruments`  
`static method`), 364 `MOT_DC_PIDParameters` (class in  
`msl.equipment.resources.thorlabs.kinesis.structs`),

`mono_start_jog()` (`msl.equipment.resources.thorlabs.kinesis.structs`),  
(`msl.equipment.resources.princeton_instruments.arc_566` `msl.instrument.PrincetonInstruments`  
`method`), 364 `MOT_DCMotor` (`msl.equipment.resources.thorlabs.kinesis.enums.MO`  
`attribute`), 419

`mono_start_scan_to_rm()` (`msl.equipment.resources.princeton_instruments.arc_423` `msl.instrument.PrincetonInstruments`  
`method`), 364 `MOT_DirectionSense` (class in  
`msl.equipment.resources.thorlabs.kinesis.enums`),

`mono_stop_jog()` (`msl.equipment.resources.thorlabs.kinesis.enums`),  
(`msl.equipment.resources.princeton_instruments.arc_423` `msl.instrument.PrincetonInstruments`  
`method`), 364 `MOT_ForwardLimitSwitch` (class in  
`msl.equipment.resources.thorlabs.kinesis.enums.MOT_H`  
`attribute`), 420

`Month` (`msl.equipment.resources.nkt.nktpdll.DateTimeType` `attribute`), 153 `MOT_Forwards` (`msl.equipment.resources.thorlabs.kinesis.enums.M`  
`attribute`), 420 `MOT_HomingParameters` (class in  
`msl.equipment.resources.thorlabs.kinesis.structs`), 559

`MOT_Backwards` (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_420` `MOT_DirectionSense`  
`attribute`), 420 `MOT_HomeLimitSwitchDirection` (class in  
`msl.equipment.resources.thorlabs.kinesis.enums`),

`MOT_BrushlessCurrentLoopParameters` (`msl.equipment.resources.thorlabs.kinesis.enums`),  
(class in `420`  
`msl.equipment.resources.thorlabs.kinesis.structs`), 559 `MOT_HomingParameters` (class in  
`msl.equipment.resources.thorlabs.kinesis.structs`), 556

`MOT_BrushlessElectricOutputParameters` (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_Immediate` (`msl.equipment.resources.thorlabs.kinesis.enums.`  
`msl.equipment.resources.thorlabs.kinesis.structs`), `attribute`), 421  
560 `MOT_JogMode` (`msl.equipment.resources.thorlabs.kinesis.enums.MO`  
`attribute`), 421

`MOT_BrushlessMotor` (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_JogModeMotorTypes` (class in  
`attribute`), 419 `msl.equipment.resources.thorlabs.kinesis.enums`),

`MOT_BrushlessPositionLoopParameters` (`msl.equipment.resources.thorlabs.kinesis.enums`),  
(class in `MOT_JogModeUndefined`  
`msl.equipment.resources.thorlabs.kinesis.structs`), (`msl.equipment.resources.thorlabs.kinesis.enums.MOT_Jo`  
`attribute`), 421  
558

`MOT_BrushlessTrackSettleParameters` `MOT_JogParameters` (class in  
(class in `msl.equipment.resources.thorlabs.kinesis.structs`),  
`msl.equipment.resources.thorlabs.kinesis.structs`), 556  
559 `MOT_JoystickParameters` (class in  
`msl.equipment.resources.thorlabs.kinesis.structs`), 558

`MOT_ButtonModes` (class in `msl.equipment.resources.thorlabs.kinesis.structs`),  
`msl.equipment.resources.thorlabs.kinesis.enums`), 421 `MOT_LimitsSoftwareApproachPolicy`

`MOT_ButtonModeUndefined` (class in  
(`msl.equipment.resources.thorlabs.kinesis.enums.MOT_ButtonModes` (`msl.equipment.resources.thorlabs.kinesis.enums`),  
`attribute`), 421 423

`MOT_ButtonParameters` (class in `MOT_LimitSwitchBreakOnContact`

(*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchBreakOnContactSwapped (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchBreakOnHome (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchBreakOnHomeSwapped (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchDirectionUndefined (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 420

MOT\_LimitSwitchIgnored (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 423

MOT\_LimitSwitchIgnored\_Rotational (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 423

MOT\_LimitSwitchIgnoreSwitch (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchIgnoreSwitchSwapped (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchMakeOnContact (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchMakeOnContactSwapped (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchMakeOnHome (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchMakeOnHomeSwapped (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchModes (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 422

MOT\_LimitSwitchModeUndefined (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 422

MOT\_LimitSwitchParameters (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 560

MOT\_LimitSwitchStopImmediate (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 423

MOT\_LimitSwitchStopImmediate\_Rotational (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 423

MOT\_LimitSwitchStopProfiled (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 423

MOT\_LimitSwitchStopProfiled\_Rotational (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes* attribute), 423

MOT\_LimitSwitchSWModes (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 422

MOT\_LimitSwitchSWModeUndefined (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchSWModes* attribute), 423

MOT\_Linear (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchSWModes* attribute), 423

MOT\_MotorTypes (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 423

MOT\_MovementDirections (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 423

MOT\_MovementModes (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 423

MOT\_Normal (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MovementModes* attribute), 420

MOT\_OpenLoop (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MovementModes* attribute), 419

MOT\_PhaseA (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MovementModes* attribute), 424

MOT\_PhaseAB (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_MovementModes* attribute), 424

MOT\_PID\_LoopMode (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 425

MOT\_PID\_ClosedLoopMode (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 425

MOT\_PIDLoopEncoderParams (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 565

MOT\_PIDLoopModeDisabled (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PID\_LoopMode* attribute), 425

MOT\_PIDOpenLoopMode (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PID\_LoopMode* attribute), 425

MOT\_PMD\_Reserved (*msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PMD\_Reserved* attribute), 425



(msl.equipment.resources.thorlabs.kinesis.enums.MOT\_LimitSwitchModes  
 attribute), 422 MOT\_TravelModeUndefined  
 MOT\_PotentiometerStep (class in (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_TravelModeUndefined), 419  
 msl.equipment.resources.thorlabs.kinesis.structs), attribute), 419  
 567 MOT\_VelocityParameters (class in  
 MOT\_PotentiometerSteps (class in msl.equipment.resources.thorlabs.kinesis.structs), 556  
 msl.equipment.resources.thorlabs.kinesis.structs), 567  
 MOT\_VelocityProfileModes (class in  
 MOT\_PowerParameters (class in msl.equipment.resources.thorlabs.kinesis.enums),  
 msl.equipment.resources.thorlabs.kinesis.structs), 421  
 561 MOT\_VelocityProfileParameters (class in  
 MOT\_Preset (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_PotentiometerSteps),  
 attribute), 421 557  
 MOT\_Profined (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_StopModes), 421  
 attribute), 421 msl.equipment.resources.thorlabs.kinesis.motion\_control)  
 MOT\_Reverse (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_TravelDirection), 420  
 attribute), 420 MotionControlCallback (in module  
 MOT\_ReverseLimitSwitch msl.equipment.resources.thorlabs.kinesis.callbacks),  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_HomeLimitSwitchDirection  
 attribute), 420 motorSignalBias  
 MOT\_Rotational (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_B  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_TravelModes  
 attribute), 419 motorSignalLimit  
 MOT\_SCurve (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_VelocityProfileModes), 422  
 attribute), 422 msl.equipment.resources.thorlabs.kinesis.structs.MOT\_B  
 attribute), 560  
 MOT\_SingleStep motorType (msl.equipment.resources.thorlabs.kinesis.structs.TLI  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_TrapzoidalModes), 421  
 attribute), 421 move() (msl.equipment.resources.optosigma.shot702.SHOT702  
 move(), 192  
 MOT\_StageAxisParameters (class in method), 192  
 msl.equipment.resources.thorlabs.kinesis.structs.move\_absolute()  
 557 (msl.equipment.resources.optosigma.shot702.SHOT702  
 MOT\_StepperMotor method), 192  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_StopModes)  
 attribute), 419 (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepp  
 MOT\_StopModes (class in method), 402  
 msl.equipment.resources.thorlabs.kinesis.enums.move\_absolute()  
 421 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepp  
 MOT\_StopModeUndefined method), 477  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_StopModes  
 attribute), 421 (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo  
 MOT\_Trapezoidal method), 500  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_VelocityProfileModes  
 attribute), 422 (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_  
 MOT\_TravelDirection (class in method), 534  
 msl.equipment.resources.thorlabs.kinesis.enums.move\_at\_velocity()  
 420 (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepp  
 MOT\_TravelDirectionDisabled method), 402  
 (msl.equipment.resources.thorlabs.kinesis.enums.MOT\_TravelDirection  
 attribute), 420 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepp  
 MOT\_TravelModes (class in method), 478  
 msl.equipment.resources.thorlabs.kinesis.enums.move\_at\_velocity()

(msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo  
 method), 500  
 move\_at\_velocity() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor  
 method), 534  
 move\_jog() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.BenchtopStepperMotors  
 method), 402  
 move\_jog() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors  
 method), 478  
 move\_jog() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo  
 method), 501  
 move\_jog() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor  
 method), 535  
 move\_relative() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo  
 method), 501  
 move\_relative() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor  
 method), 535  
 move\_relative() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.BenchtopStepperMotor  
 method), 403  
 move\_relative() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors  
 method), 478  
 move\_relative() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo  
 method), 501  
 move\_relative() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor  
 method), 535  
 move\_relative\_distance() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motors.BenchtopStepperMotor  
 method), 403  
 move\_relative\_distance() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors  
 method), 478  
 move\_relative\_distance() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo  
 method), 501  
 move\_relative\_distance() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor  
 method), 535  
 move\_to\_position() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor  
 method), 403  
 move\_to\_position() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motors.IntegratedStepperMotors  
 method), 478  
 move\_to\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.KCubeDCServo  
 method), 501  
 move\_to\_position() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor  
 method), 535

- module, [32](#)
- `msl.equipment.connection_serial`
  - module, [33](#)
- `msl.equipment.connection_socket`
  - module, [35](#)
- `msl.equipment.connection_tcpip_hislip`
  - module, [36](#)
- `msl.equipment.connection_tcpip_vx11`
  - module, [39](#)
- `msl.equipment.connection_zeromq`
  - module, [42](#)
- `msl.equipment.constants`
  - module, [43](#)
- `msl.equipment.database`
  - module, [45](#)
- `msl.equipment.dns_service_discovery`
  - module, [47](#)
- `msl.equipment.exceptions`
  - module, [48](#)
- `msl.equipment.factory`
  - module, [50](#)
- `msl.equipment.hislip`
  - module, [50](#)
- `msl.equipment.resources`
  - module, [81](#)
- `msl.equipment.resources.aim_tti`
  - module, [84](#)
- `msl.equipment.resources.aim_tti.mx_series`
  - module, [84](#)
- `msl.equipment.resources.avantes`
  - module, [91](#)
- `msl.equipment.resources.avantes.avaspec`
  - module, [91](#)
- `msl.equipment.resources.bentham`
  - module, [121](#)
- `msl.equipment.resources.bentham.benhw32`
  - module, [122](#)
- `msl.equipment.resources.bentham.benhw64`
  - module, [124](#)
- `msl.equipment.resources.bentham.errors`
  - module, [125](#)
- `msl.equipment.resources.bentham.tokens`
  - module, [125](#)
- `msl.equipment.resources.cmi`
  - module, [125](#)
- `msl.equipment.resources.cmi.sia3`
  - module, [125](#)
- `msl.equipment.resources.dataray`
  - module, [127](#)
- `msl.equipment.resources.dataray.datarayocx_32`
  - module, [127](#)
- `msl.equipment.resources.dataray.datarayocx_64`
  - module, [127](#)
- `msl.equipment.resources.dmm`
  - module, [81](#)
- `msl.equipment.resources.electron_dynamics`
  - module, [130](#)
- `msl.equipment.resources.electron_dynamics.tc_series`
  - module, [130](#)
- `msl.equipment.resources.energetiq`
  - module, [135](#)
- `msl.equipment.resources.energetiq.eq99`
  - module, [135](#)
- `msl.equipment.resources.mks_instruments`
  - module, [141](#)
- `msl.equipment.resources.mks_instruments.pr4000b`
  - module, [141](#)
- `msl.equipment.resources.nkt`
  - module, [153](#)
- `msl.equipment.resources.nkt.nktpdll`
  - module, [153](#)
- `msl.equipment.resources.omega`
  - module, [186](#)
- `msl.equipment.resources.omega.ithx`
  - module, [186](#)
- `msl.equipment.resources.optosigma`
  - module, [190](#)
- `msl.equipment.resources.optosigma.shot702`
  - module, [190](#)
- `msl.equipment.resources.optronic_laboratories`
  - module, [197](#)
- `msl.equipment.resources.optronic_laboratories.ol756`
  - module, [197](#)
- `msl.equipment.resources.optronic_laboratories.ol756`
  - module, [215](#)
- `msl.equipment.resources.optronic_laboratories.ol_cu`
  - module, [216](#)
- `msl.equipment.resources.picotech`
  - module, [219](#)
- `msl.equipment.resources.picotech.errors`
  - module, [219](#)
- `msl.equipment.resources.picotech.picoscope`
  - module, [220](#)
- `msl.equipment.resources.picotech.picoscope.callback`
  - module, [220](#)
- `msl.equipment.resources.picotech.picoscope.channel`
  - module, [222](#)
- `msl.equipment.resources.picotech.picoscope.enums`
  - module, [224](#)
- `msl.equipment.resources.picotech.picoscope.function`
  - module, [310](#)
- `msl.equipment.resources.picotech.picoscope.helper`



Index 677





|                                  |   |   |
|----------------------------------|---|---|
| <code>notchFilterEnabled</code>  | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.structs.QDcNotchFilterParameters</code><br>attribute), 574 | NR_WAVELEN_POL_COEF   |
| <code>NotchFilterOff</code>      | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.enums.PPcAttNotchFilterState</code><br>attribute), 434     | NS (msl.equipment.resources.picotech.picoscope.enums.PS2000ATimeNotchFilterState                    |
| <code>NotchFilterOn</code>       | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.PPcAttNotchFilterState</code><br>attribute), 434   | NS (msl.equipment.resources.picotech.picoscope.enums.PS2000TimeNotchFilterParameters                |
| <code>notchFilterQ</code>        | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.QDcNotchFilterParameters</code><br>attribute), 573   | NS (msl.equipment.resources.picotech.picoscope.enums.PS3000ATimeNotchFilterParameters               |
| <code>notchFilterQ</code>        | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.QDcNotchFilterParameters</code><br>attribute), 574   | NS (msl.equipment.resources.picotech.picoscope.enums.PS3000TimeNotchFilterParameters                |
| <code>notes</code>               | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.TLI_HardwareInformation</code><br>attribute), 556  | NS (msl.equipment.resources.picotech.picoscope.enums.PS4000ATimeNotchFilterParameters               |
| <code>notUsed</code>             | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.MOTriBrushlessCurrentLoopParameters</code><br>attribute), 560  | NS (msl.equipment.resources.picotech.picoscope.enums.PS4000TimeNotchFilterParameters                |
| <code>notUsed</code>             | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.MOTriBrushlessElectricOutputParameters</code><br>attribute), 560   | NS (msl.equipment.resources.picotech.picoscope.enums.PS5000ATimeNotchFilterParameters               |
| <code>notUsed</code>             | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.MOTriBrushlessPositionLoopParameters</code><br>attribute), 559   | NS (msl.equipment.resources.picotech.picoscope.enums.PS5000TimeNotchFilterParameters                |
| <code>notUsed</code>             | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.MOTriBrushlessTrackSettleParameters</code><br>attribute), 559  | NS (msl.equipment.resources.picotech.picoscope.enums.PS6000TimeNotchFilterParameters                |
| <code>notUsed</code>             | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.MOTriVelocityProfileParameters</code><br>attribute), 557   | NT_AbsPowerCircleMode   |
| <code>notYetInUse</code>         | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.NT_CircleSettings</code><br>attribute), 563  | NT_AbsPowerCircleMode   |
| <code>notYetInUse</code>         | ( <code>msl.equipment.resources.thorlabs.kinesis.structs.NT_PowerSettings</code><br>attribute), 577   | NT_AbsPowerCircleMode   |
| <code>NR360</code>               | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.NT_AutoRangeAtParameter</code><br>attribute), 457  | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter</code><br>attribute), 429  |
| <code>NR360</code>               | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.TST_Strategy</code><br>attribute), 461   | NT_AutoRangeAtSelected  |
| <code>NR_ANALOG_OUTPUTS</code>   | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 102   | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter</code><br>attribute), 429  |
| <code>NR_DAC_POL_COEF</code>     | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 100   | NT_BadSignal (msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter                      |
| <code>NR_DEFECTIVE_PIXELS</code> | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 100   | NT_BNC_10v (msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter                        |
| <code>NR_DIGITAL_INPUTS</code>   | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 102   | NT_BNC_1v (msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter                         |
| <code>NR_DIGITAL_OUTPUTS</code>  | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 101   | NT_BNC_2v (msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter                         |
| <code>NR_NONLIN_POL_COEF</code>  | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 100   | NT_BNC_5v (msl.equipment.resources.thorlabs.kinesis.enums.NT_TIAAtParameter                         |
| <code>NR_TEMP_POL_COEF</code>    | ( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>attribute), 100   | NT_BNCModeLVOut   |
|                                  |   | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.BNT_BNCModeLVOut</code><br>attribute), 432   |
|                                  |   | NT_BNCModeTrigger   |
|                                  |   | ( <code>msl.equipment.resources.thorlabs.kinesis.enums.BNT_BNCModeTrigger</code><br>attribute), 432 |
|                                  |   | NT_CircleAdjustment (class in   |



`msl.equipment.resources.thorlabs.kinesis.enums)`, `(msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSignalSelection`  
 428 `attribute)`, 426  
`NT_CircleDiameterLUT` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FullRange`  
`msl.equipment.resources.thorlabs.kinesis.structs)`, `attribute)`, 449  
 562  
`NT_CircleDiameterMode` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FullRange`  
`msl.equipment.resources.thorlabs.kinesis.enums.NT_FullRange` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
 428 `attribute)`, 430  
`NT_CircleParameters` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FullRange`  
`msl.equipment.resources.thorlabs.kinesis.structs)`, `attribute)`, 563  
 562  
`NT_ClosedLoop` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_ClosedLoop` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 426  
`NT_ClosedLoopSmoothed` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_ClosedLoopSmoothed` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 426  
`NT_ControlMode` (class in `NT_HVComponent` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_ControlMode` (class in  
`msl.equipment.resources.thorlabs.kinesis.structs)`, `attribute)`, 425  
 561  
`NT_ControlModeUndefined` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_ControlModeUndefined` (class in `NT_HVComponent` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 426  
`NT_CubeCircleAdjustment` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_CubeCircleAdjustment` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 428  
`NT_CurrentLimit_100mA` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_CurrentLimit_100mA` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 431  
`NT_CurrentLimit_250mA` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_CurrentLimit_250mA` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 431  
`NT_CurrentLimit_500mA` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_CurrentLimit_500mA` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 431  
`NT_Db` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_Db` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 430  
`NT_Even` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_Even` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 427  
`NT_FeedbackSignalAC` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSignalAC` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 431  
`NT_FeedbackSignalDC` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSignalDC` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 431  
`NT_FeedbackSource` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSource` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`msl.equipment.resources.thorlabs.kinesis.enums)`, `attribute)`, 429  
 426  
`NT_FeedbackSourceUndefined` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSourceUndefined` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 446  
`NT_FeedbackSourceUndefined` (`msl.equipment.resources.thorlabs.kinesis.enums.NT_FeedbackSourceUndefined` (class in `NT_FullRange` (`msl.equipment.resources.thorlabs.kinesis.structs)`,  
`attribute)`, 446

attribute), 429

NT\_LowPassFilterParameters (class in NT\_OverRange (msl.equipment.resources.thorlabs.kinesis.enums.N  
msl.equipment.resources.thorlabs.kinesis.structs), attribute), 428

NT\_ParameterCircleMode 562

NT\_LowPassFrequency (class in (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Circ  
msl.equipment.resources.thorlabs.kinesis.enums), attribute), 428

NT\_Piezo (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Mo  
attribute), 425

NT\_LowPassNone (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Piezo), 425

NT\_RangeAtParameter (class in NT\_LowPassFrequency (class in  
msl.equipment.resources.thorlabs.kinesis.enums), attribute), 429

NT\_LUTCircleMode (msl.equipment.resources.thorlabs.kinesis.enums.NT\_SquareCircleAdjustment  
attribute), 428

NT\_ManualRangeAtParameter (msl.equipment.resources.thorlabs.kinesis.enums.NT\_SquareCircleAdjustment  
attribute), 425

NT\_ManualRangeAtSelected (msl.equipment.resources.thorlabs.kinesis.enums.NT\_SquareCircleAdjustment  
attribute), 430

NT\_Mode (class in NT\_SquareCircleAdjustment  
msl.equipment.resources.thorlabs.kinesis.enums), (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Circ  
attribute), 425

NT\_ModeUndefined NT\_TIA (msl.equipment.resources.thorlabs.kinesis.enums.KNA\_Fee  
attribute), 425

NT\_Odd (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Mode), 426

NT\_OddAndEven (msl.equipment.resources.thorlabs.kinesis.enums.NT\_Odd), 426

NT\_OddOrEven (class in NT\_TIARange (class in  
msl.equipment.resources.thorlabs.kinesis.enums), (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIA  
attribute), 427

NT\_OpenLoop (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OddOrEven), 426

NT\_OpenLoopSmoothed (msl.equipment.resources.thorlabs.kinesis.enums.NT\_TIARange), 427

NT\_OutputFilter\_100Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OpenLoopSmoothed  
attribute), 427

NT\_OutputFilter\_10Hz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OutputFilter\_100Hz  
attribute), 431

NT\_OutputFilter\_5kHz (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OutputFilter\_10Hz  
attribute), 431

NT\_OutputFilter\_None (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OutputFilter\_5kHz  
attribute), 426

NT\_OutputVoltageRoute (class in NT\_OutputFilter\_None  
msl.equipment.resources.thorlabs.kinesis.enums), attribute), 427

NT\_TIARange3\_30nA (msl.equipment.resources.thorlabs.kinesis.enums.NT\_OutputVoltageRoute  
attribute), 427

(*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange4* attribute), 427

NT\_TTIRange4\_100nA (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange4* attribute), 427

NT\_TTIRange5\_300nA (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange5* attribute), 427

NT\_TTIRange6\_1uA (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange6* attribute), 427

NT\_TTIRange7\_3uA (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange7* attribute), 427

NT\_TTIRange8\_10uA (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange8* attribute), 427

NT\_TTIRange9\_30uA (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRange9* attribute), 427

NT\_TTIRangeMode (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 428

NT\_TTIRangeModeUndefined (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_TTIRangeMode* attribute), 429

NT\_TTIRangeParameters (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 562

NT\_TTIReading (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 563

NT\_Tracking (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_Tracking* attribute), 425

NT\_UnderOrOver (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 427

NT\_UnderRange (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_UnderOrOver* attribute), 428

NT\_UserDefined (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_UserDefined* attribute), 449

NT\_UserDefined (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_UserDefined* attribute), 430

NT\_VerticalTracking (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_VerticalTracking* attribute), 425

NT\_Voltage (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_Voltage* attribute), 449

NT\_Voltage (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_Voltage* attribute), 430

NT\_VoltageRange (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 429

NT\_VoltageRange\_10v (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_VoltageRange* attribute), 429

NT\_VoltageRange\_5v (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_VoltageRange* attribute), 429

NT\_VoltageRangeUndefined (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_VoltageRange* attribute), 429

NT\_Watts (*msl.equipment.resources.thorlabs.kinesis.enums.NT\_Watts* attribute), 430

NTC1\_ID (*msl.equipment.resources.avantes.avaspec.Avantes* attribute), 102

NTC2\_ID (*msl.equipment.resources.avantes.avaspec.Avantes* attribute), 102

NULL (*msl.equipment.vxi11.OperationFlag* attribute), 586

num\_locks (*msl.equipment.hislip.AsyncLockInfoResponse* property), 57

num\_samples (*msl.equipment.resources.picotech.picoscope.channels* attribute), 204

numChannels (*msl.equipment.resources.thorlabs.kinesis.structs.TL\_TTIParameters* attribute), 556

numCycles (*msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT* attribute), 564

numCycles (*msl.equipment.resources.thorlabs.kinesis.structs.SC\_C* attribute), 576

Numerator (*msl.equipment.resources.nkt.nktpdll.ParameterSetType* attribute), 54

numOutTriggerRepeat (*msl.equipment.resources.thorlabs.kinesis.structs.PZ\_LUT* attribute), 564

ODD (*msl.equipment.constants.Parity* attribute), 44

OEM\_DATA\_LEN (*msl.equipment.resources.avantes.avaspec.Avantes* attribute), 100

OemDataType (class in *msl.equipment.resources.avantes.avaspec*), 98

OFF (*msl.equipment.resources.picotech.picoscope.enums.PS2000A* attribute), 235

OFF (*msl.equipment.resources.picotech.picoscope.enums.PS2000A* attribute), 237

OFF (*msl.equipment.resources.picotech.picoscope.enums.PS2000Et* attribute), 227

OFF (*msl.equipment.resources.picotech.picoscope.enums.PS3000A* attribute), 227

- attribute), 252
- attribute), 578
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS3000AExtOp method), 388
- attribute), 254
- method), 388
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS3000AExtOp method), 388
- attribute), 245
- 44
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS4000AFuncLe attribute), 275
- (msl.equipment.constants.StopBits
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS4000AFuncLe attribute), 273
- OP\_200Hz (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_1
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS4000AFuncLe attribute), 269
- OP\_Unfiltered (msl.equipment.resources.thorlabs.kinesis.enums.
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS4000AFuncLe attribute), 263
- open() (msl.equipment.resources.picotech.pt104.PT104
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS5000AFuncLe attribute), 295
- open() (msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX2
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS5000AFuncLe attribute), 293
- open() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS5000AFuncLe attribute), 287
- open() (msl.equipment.resources.thorlabs.kinesis.filter\_flipper.Filt
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS6000AFuncLe attribute), 304
- open() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepp
- OFF (msl.equipment.resources.picotech.picoscope.enums.PS6000AFuncLe attribute), 306
- open() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo
- OFF (msl.equipment.resources.picotech.pt104.Pt104DataType method), 501
- attribute), 358
- open() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoid.
- OFF (msl.equipment.resources.thorlabs.fwx2c.SensorMode method), 519
- attribute), 578
- open() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_r
- Off (msl.equipment.resources.thorlabs.kinesis.enums.PPC\_Display attribute), 436
- attribute), 436
- open\_filter() (msl.equipment.resources.princeton\_instruments.
- off() (msl.equipment.resources.raicol.raicol\_tec.RaicolTEC method), 365
- method), 388
- open\_filter\_port()
- Offset (msl.equipment.resources.nkt.nktpdll.ParameterSetType attribute), 154
- method), 365
- offsetDistance
- open\_filter\_serial()
- (msl.equipment.resources.thorlabs.kinesis.structs.MQTTsHequippRatures.nces.princeton\_instruments.arc\_instr
- attribute), 557
- method), 365
- OK (msl.equipment.resources.picotech.errors.PS2000OpenMono method), 365
- attribute), 219
- method), 365
- OK (msl.equipment.resources.picotech.errors.PS3000OpenMonoPort method), 365
- attribute), 220
- (msl.equipment.resources.princeton\_instruments.arc\_instr
- OL756 (class in method), 365
- msl.equipment.resources.optronic\_laboratories.openMonoSerial()
- 197
- (msl.equipment.resources.princeton\_instruments.arc\_instr
- OL756 (class in method), 365
- msl.equipment.resources.optronic\_laboratories.openPorts(64)
- 215
- (msl.equipment.resources.nkt.nktpdll.NKT
- static method), 169
- OLCurrentSource (class in open\_readout()
- msl.equipment.resources.optronic\_laboratories.ol\_current\_source
- 216
- (msl.equipment.resources.princeton\_instruments.arc\_instr
- method), 365
- OmegaError, 49
- open\_readout\_port()
- ON (msl.equipment.resources.thorlabs.fwx2c.SensorMode (msl.equipment.resources.princeton\_instruments.arc\_instr



[method](#)), 366  
[open\\_unit\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_2k3k](#) attribute), 253  
[method](#)), 317  
[open\\_unit\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_4k](#) attribute), 253  
[method](#)), 323  
[open\\_unit\\_async\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_2k3k](#) attribute), 281  
[method](#)), 317  
[open\\_unit\\_async\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_4k](#) attribute), 268  
[method](#)), 323  
[open\\_unit\\_async\\_ex\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_4k](#) attribute), 292  
[method](#)), 337  
[open\\_unit\\_ex\(\)](#) ([msl.equipment.resources.thorlabs.fwx2c.TriggerMode](#) attribute), 400  
[method](#)), 338  
[open\\_unit\\_progress\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_2k3k](#) attribute), 240  
[method](#)), 317  
[open\\_unit\\_progress\(\)](#) ([msl.equipment.resources.picotech.picoscope.picoscope\\_4k](#) attribute), 229  
[method](#)), 323  
[open\\_via\\_ip\(\)](#) ([msl.equipment.resources.picotech.pt104.PT104](#) attribute), 246  
[method](#)), 360  
[openLoopOption](#) ([msl.equipment.resources.thorlabs.kinesis.structs.PZ\\_LUT](#) attribute), 278  
[attribute](#)), 574  
[openTime](#) ([msl.equipment.resources.thorlabs.kinesis.structs.PZ\\_LUT](#) attribute), 299  
[OperationFlag](#) (class in [msl.equipment.vxi11](#)), 586  
[Optical](#) ([msl.equipment.resources.thorlabs.kinesis.structs.PZ\\_LUT](#) attribute), 435  
[OPTICAL\\_SWITCH](#) ([msl.equipment.resources.picotech.enums.PS4000](#) attribute), 262  
[OptoSigmaError](#), 49  
[OptronicLaboratoriesError](#), 49  
[OR](#) ([msl.equipment.resources.picotech.enums.PS2000A](#) attribute), 232  
[OS\\_NOT\\_SUPPORTED](#) ([msl.equipment.resources.picotech.errors.PS3000Error](#) attribute), 219  
[OS\\_NOT\\_SUPPORTED](#) ([msl.equipment.resources.picotech.errors.PS3000Error](#) attribute), 220  
[OUT\\_OF\\_RANGE](#) ([msl.equipment.resources.picotech.enums.PS2000A](#) attribute), 241  
[OUT\\_OF\\_RANGE](#) ([msl.equipment.resources.picotech.enums.PS2000A](#) attribute), 230

`msl.equipment.utils`), 586

`parse_pico_scope_api_header()` (in module `msl.equipment.resources.picotech.picoscope.picoscope_2k`)

[316](#) [method](#)), [313](#)  
**PicoScope3000** (class in `PLS_X25MM` (`msl.equipment.resources.thorlabs.kinesis.enums.KST_`  
`msl.equipment.resources.picotech.picoscope.ps3000`), [457](#)  
[332](#) `PLS_X25MM` (`msl.equipment.resources.thorlabs.kinesis.enums.TST_`  
**PicoScope3000A** (class in `attribute`), [461](#)  
`msl.equipment.resources.picotech.picoscope.ps3000A` `PLS_X25MM_HiRes`  
[334](#) (`msl.equipment.resources.thorlabs.kinesis.enums.KST_`  
**PicoScope4000** (class in `attribute`), [457](#)  
`msl.equipment.resources.picotech.picoscope.ps4000` `PLS_X25MM_HiRes`  
[336](#) (`msl.equipment.resources.thorlabs.kinesis.enums.TST_`  
**PicoScope4000A** (class in `attribute`), [461](#)  
`msl.equipment.resources.picotech.picoscope.ps4000A` `point_port_add()`  
[338](#) (`msl.equipment.resources.nkt.nktpdll.NKT`  
**PicoScope5000** (class in `method`), [170](#)  
`msl.equipment.resources.picotech.picoscope.ps5000` `point_port_del()`  
[340](#) (`msl.equipment.resources.nkt.nktpdll.NKT`  
**PicoScope5000A** (class in `method`), [170](#)  
`msl.equipment.resources.picotech.picoscope.ps5000A` `point_port_get()`  
[341](#) (`msl.equipment.resources.nkt.nktpdll.NKT`  
**PicoScope6000** (class in `method`), [170](#)  
`msl.equipment.resources.picotech.picoscope.ps6000` `polarity1` (`msl.equipment.resources.thorlabs.kinesis.structs.KLD_`  
[343](#) `attribute`), [569](#)  
**PicoScopeApi** (class in `polarity1` (`msl.equipment.resources.thorlabs.kinesis.structs.KLS_`  
`msl.equipment.resources.picotech.picoscope.picoscopeapi`), [570](#)  
[318](#) `polarity2` (`msl.equipment.resources.thorlabs.kinesis.structs.KLD_`  
**PicoScopeChannel** (class in `attribute`), [569](#)  
`msl.equipment.resources.picotech.picoscope.picoscopechannel` `polarity2` (`msl.equipment.resources.thorlabs.kinesis.structs.KLS_`  
[222](#) `attribute`), [570](#)  
**PicoScopeInfoApi** (class in `poll_scan()` (`msl.equipment.resources.avantes.avaspec.Avantes`  
`msl.equipment.resources.picotech.picoscope.enums`), [117](#)  
[224](#) `polling_duration()`  
**PicoTechError**, [49](#) (`msl.equipment.resources.thorlabs.kinesis.benchtop_stepp`  
**PID** (`msl.equipment.resources.thorlabs.kinesis.structs.TLI_DemioInfo`), [404](#)  
`attribute`), [555](#) `polling_duration()`  
**PIDConstsD** (`msl.equipment.resources.thorlabs.kinesis.structs.PIDConstsD`), [466](#)  
`attribute`), [564](#) `method`), [466](#)  
**PIDConstsDFc** (`msl.equipment.resources.thorlabs.kinesis.structs.PIDConstsDFc`), [564](#)  
`attribute`), [564](#) (`msl.equipment.resources.thorlabs.kinesis.integrated_step`  
**PIDConstsI** (`msl.equipment.resources.thorlabs.kinesis.structs.PIDConstsI`), [564](#)  
`attribute`), [564](#) `polling_duration()`  
**PIDConstsP** (`msl.equipment.resources.thorlabs.kinesis.structs.PIDConstsP`), [502](#)  
`attribute`), [564](#) (`msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo`  
`method`), [502](#)  
**PIDDerivFilterOn** (`msl.equipment.resources.thorlabs.kinesis.structs.PPIDConsts`), [519](#)  
`attribute`), [564](#) `method`), [519](#)  
**PIDOutputLimit** (`msl.equipment.resources.thorlabs.kinesis.structs.MOT_PIDOutputLimit`), [536](#)  
`attribute`), [565](#) `method`), [536](#)  
**PIDTolerance** (`msl.equipment.resources.thorlabs.kinesis.structs.MOT_PIDTolerance`), [36](#)  
`attribute), 565 property), 36  
ping_unit() (msl.equipment.resources.picotech.picoscope.picoscopeapi), 36  
msl.equipment.resources.thorlabs.kinesis.kcube_stepper_`





attribute), 256  
 POSITIVE\_RUNT (*msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection*  
 attribute), 278  
 POSITIVE\_RUNT (*msl.equipment.resources.picotech.picoscope.enums.PS2000AThresholdDirection*  
 attribute), 267  
 POSITIVE\_RUNT (*msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection*  
 attribute), 299  
 POSITIVE\_RUNT (*msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection*  
 attribute), 309  
 PosRaw (*msl.equipment.resources.thorlabs.kinesis.enums.PPC\_IOWaveParametersMode*  
 attribute), 435  
 postCycleDelay (attribute), 293  
 (msl.equipment.resources.thorlabs.kinesis.structs.MQTTsPotentiometerSteps, *msl.equipment.resources.picotech.picoscope.enums.PS6000BThresholdDirection*  
 attribute), 564  
 potentiometerStepParameters PRBS\_MAX\_FREQUENCY  
 (msl.equipment.resources.thorlabs.kinesis.structs.MQTTsPotentiometerSteps, *msl.equipment.resources.picotech.picoscope.ps2000a.PicotechPotentiometerSteps*  
 attribute), 567  
 PPC\_DerivFilterState (class in PRBS\_MAX\_FREQUENCY  
*msl.equipment.resources.thorlabs.kinesis.enums*), (msl.equipment.resources.picotech.picoscope.ps3000a.PicotechPotentiometerSteps  
 attribute), 433  
 PPC\_DisplayIntensity (class in PRBS\_MAX\_FREQUENCY  
*msl.equipment.resources.thorlabs.kinesis.enums*), (msl.equipment.resources.picotech.picoscope.ps6000.PicotechPotentiometerSteps  
 attribute), 436  
 PPC\_IOControlMode (class in PRBS\_MIN\_FREQUENCY  
*msl.equipment.resources.thorlabs.kinesis.enums*), (msl.equipment.resources.picotech.picoscope.ps2000a.PicotechPotentiometerSteps  
 attribute), 434  
 PPC\_IOFeedbackSourceDefinition (class in PRBS\_MIN\_FREQUENCY  
*msl.equipment.resources.thorlabs.kinesis.enums*), (msl.equipment.resources.picotech.picoscope.ps3000a.PicotechPotentiometerSteps  
 attribute), 435  
 PPC\_IIOOutputBandwidth (class in pre\_trigger (msl.equipment.resources.picotech.picoscope.picoscope.enums),  
*msl.equipment.resources.thorlabs.kinesis.enums*, property), 435  
 PPC\_IIOOutputMode (class in attribute), 564  
*msl.equipment.resources.thorlabs.kinesis.enums*, prepare\_measure()  
 (msl.equipment.resources.avantes.avaspec.Avantes  
 attribute), 435  
 PPC\_IOSettings (class in method), 117  
*msl.equipment.resources.thorlabs.kinesis.structs.KM2000*, PresetPos1 (msl.equipment.resources.thorlabs.kinesis.structs.KM2000  
 attribute), 565  
 PPC\_NotchFilterChannel (class in PresetPos1 (msl.equipment.resources.thorlabs.kinesis.structs.KM2000  
*msl.equipment.resources.thorlabs.kinesis.enums*), attribute), 572  
 434 PresetPos2 (msl.equipment.resources.thorlabs.kinesis.structs.KM2000  
 attribute), 567  
 PPC\_NotchFilterState (class in attribute), 567  
*msl.equipment.resources.thorlabs.kinesis.enums*, PresetPos2 (msl.equipment.resources.thorlabs.kinesis.structs.KM2000  
 attribute), 433  
 PPC\_NotchParams (class in PRESSURE\_SENSOR\_50BAR  
*msl.equipment.resources.thorlabs.kinesis.structs*), (msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection  
 attribute), 565  
 PPC\_PIDConsts (class in PRESSURE\_SENSOR\_5BAR  
*msl.equipment.resources.thorlabs.kinesis.structs*), (msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection  
 attribute), 564  
 PR4000B (class in PrincetonInstruments (class in

[msl.equipment.resources.princeton\\_instruments.arc\\_instrument](#), 578  
[361](#) [proportionalTerm](#)  
[PrincetonInstrumentsError](#), 49 [\(msl.equipment.resources.thorlabs.kinesis.structs.KNA\\_Fe](#)  
[print\\_class\\_def\\_signatures\(\)](#) (in module [attribute](#)), 572  
[msl.equipment.resources.picotech.picoscope.proportionalTerm](#)  
[311](#) [\(msl.equipment.resources.thorlabs.kinesis.structs.PZ\\_Fe](#)  
[print\\_common\\_functions\(\)](#) (in module [attribute](#)), 564  
[msl.equipment.resources.picotech.picoscope.proportional\\_version](#)  
[311](#) [\(msl.equipment.hislip.InitializeResponse](#)  
[print\\_define\\_statements\(\)](#) (in module [property](#)), 55  
[msl.equipment.resources.picotech.picoscope.PS2000A](#) [\(msl.equipment.resources.picotech.picoscope.enums.PS2000ATi](#)  
[311](#) [attribute](#)), 236  
[print\\_resources\(\)](#) (in module [msl.equipment](#)), [PS \(msl.equipment.resources.picotech.picoscope.enums.PS2000Tim](#)  
[17](#) [attribute](#)), 226  
[PROBES](#) [\(msl.equipment.resources.picotech.picoscope.PS4000A](#) [\(msl.equipment.resources.picotech.picoscope.enums.PS3000ATi](#)  
[attribute](#)), 262 [attribute](#)), 252  
[PROC\\_UNAVAIL](#) [\(msl.equipment.vxi11.AcceptStatus](#) [PS \(msl.equipment.resources.picotech.picoscope.enums.PS3000Tim](#)  
[attribute](#)), 587 [attribute](#)), 244  
[ProcessControlType](#) (class in [PS \(msl.equipment.resources.picotech.picoscope.enums.PS4000ATi](#)  
[msl.equipment.resources.avantes.avaspec](#)), [attribute](#)), 274  
[98](#) [PS \(msl.equipment.resources.picotech.picoscope.enums.PS4000Tim](#)  
[PROG\\_MISMATCH](#) [\(msl.equipment.vxi11.AcceptStatus](#) [attribute](#)), 263  
[attribute](#)), 587 [PS \(msl.equipment.resources.picotech.picoscope.enums.PS5000ATi](#)  
[PROG\\_UNAVAIL](#) [\(msl.equipment.vxi11.AcceptStatus](#) [attribute](#)), 296  
[attribute](#)), 587 [PS \(msl.equipment.resources.picotech.picoscope.enums.PS5000Tim](#)  
[PROLOGIX](#) [\(msl.equipment.constants.Interface](#) [attribute](#)), 287  
[attribute](#)), 44 [PS \(msl.equipment.resources.picotech.picoscope.enums.PS6000Tim](#)  
[prologue](#) [\(msl.equipment.hislip.Message](#) [attribute](#)), 305  
[attribute](#)), 53 [PS2000A\\_ThresholdMode](#) (class in  
[properties](#) [\(msl.equipment.record\\_types.ConnectionRecord](#) [msl.equipment.resources.picotech.picoscope.enums](#)),  
[attribute](#)), 79 [239](#)  
[proportionalGain](#) [ps2000aBlockReady](#) (in module  
[\(msl.equipment.resources.thorlabs.kinesis.structs.MOTs/BuraphCurrenCoopPotelepicoscope.callbacks\)](#),  
[attribute](#)), 560 [220](#)  
[proportionalGain](#) [PS2000AChannel](#) (class in  
[\(msl.equipment.resources.thorlabs.kinesis.structs.MOTs/BuraphPositionLeopRatepicoscope.enums\)](#),  
[attribute](#)), 559 [231](#)  
[proportionalGain](#) [PS2000AChannelBufferIndex](#) (class in  
[\(msl.equipment.resources.thorlabs.kinesis.structs.MOTs/IDQuiPileParasaterspicoscope.enums\)](#),  
[attribute](#)), 561 [230](#)  
[proportionalGain](#) [PS2000AChannelInfo](#) (class in  
[\(msl.equipment.resources.thorlabs.kinesis.structs.MOTs/PLDmpEnraderPaspicoscope.enums\)](#),  
[attribute](#)), 566 [235](#)  
[proportionalGain](#) [PS2000ACoupling](#) (class in  
[\(msl.equipment.resources.thorlabs.kinesis.structs.QDnKlocRpmetersresources.picotech.picoscope.enums\)](#),  
[attribute](#)), 573 [235](#)  
[proportionalGain](#) [ps2000aDataReady](#) (in module  
[\(msl.equipment.resources.thorlabs.kinesis.structs.QDnFILEPipmetersresources.picotech.picoscope.callbacks\)](#),  
[attribute](#)), 573 [221](#)  
[proportionalGain](#) [PS2000ADigitalChannel](#) (class in  
[\(msl.equipment.resources.thorlabs.kinesis.structs.TCnIsobRipmetersresources.picotech.picoscope.enums\)](#),

|  |     |
|--|-----|
| 232  | 347 |
| PS2000ADigitalChannelDirections (class in PS2000ATriggerConditions (class in msl.equipment.resources.picotech.picoscope.structs), msl.equipment.resources.picotech.picoscope.structs), |     |
| 347  | 346 |
| PS2000ADigitalDirection (class in PS2000ATriggerOperand (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                |     |
| 240  | 232 |
| PS2000AEtsMode (class in PS2000ATriggerState (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                           |     |
| 235  | 240 |
| PS2000AExtraOperations (class in PS2000AWaveType (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                       |     |
| 237  | 236 |
| PS2000AHoldOffType (class in PS2000ButtonState (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                         |     |
| 241  | 227 |
| PS2000AIndexMode (class in PS2000Channel (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                               |     |
| 238  | 224 |
| PS2000APulseWidthType (class in PS2000DigitalPort (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                      |     |
| 241  | 232 |
| PS2000APwqConditions (class in PS2000Error (class in msl.equipment.resources.picotech.picoscope.structs), msl.equipment.resources.picotech.errors),                                    |     |
| 346  | 219 |
| PS2000ARange (class in PS2000EtsMode (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                                   |     |
| 234  | 227 |
| PS2000ARatioMode (class in PS2000Info (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                                  |     |
| 241  | 226 |
| PS2000ASigGenTrigSource (class in PS2000OpenProgress (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                   |     |
| 238  | 227 |
| PS2000ASigGenTrigType (class in PS2000PulseWidthType (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                   |     |
| 237  | 230 |
| ps2000aStreamingReady (in module PS2000PwqConditions (class in msl.equipment.resources.picotech.picoscope.callbacks), msl.equipment.resources.picotech.picoscope.structs),             |     |
| 221  | 345 |
| PS2000ASweepType (class in PS2000Range (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                                 |     |
| 236  | 225 |
| PS2000AThresholdDirection (class in PS2000SweepType (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                    |     |
| 239  | 228 |
| PS2000ATimeUnits (class in PS2000ThresholdDirection (class in msl.equipment.resources.picotech.picoscope.enums), msl.equipment.resources.picotech.picoscope.enums),                    |     |
| 235  | 229 |
| PS2000ATriggerChannelProperties (class in PS2000ThresholdMode (class in msl.equipment.resources.picotech.picoscope.structs), msl.equipment.resources.picotech.picoscope.enums),        |     |

229 attribute), 330

PS2000TimeUnits (class in PS2205\_MAX\_ETS\_CYCLES  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps2000.Pico  
225 attribute), 330

PS2000TriggerChannelProperties (class in PS2205\_MAX\_ETS\_INTERLEAVE  
msl.equipment.resources.picotech.picoscope.structs),(msl.equipment.resources.picotech.picoscope.ps2000.Pico  
345 attribute), 330

PS2000TriggerConditions (class in PS2206\_MAX\_ETS\_CYCLES  
msl.equipment.resources.picotech.picoscope.structs),(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
345 attribute), 331

PS2000TriggerDirection (class in PS2206\_MAX\_INTERLEAVE  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
226 attribute), 331

PS2000TriggerState (class in PS2207\_MAX\_ETS\_CYCLES  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
230 attribute), 331

PS2000WaveType (class in PS2207\_MAX\_INTERLEAVE  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps2000a.Pico  
228 attribute), 331

PS2104\_MAX\_ETS\_CYCLES PS2208\_MAX\_ETS\_CYCLES  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.ps2000a.Pico  
attribute), 329 attribute), 331

PS2104\_MAX\_ETS\_INTERLEAVE PS2208\_MAX\_INTERLEAVE  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.ps2000a.Pico  
attribute), 329 attribute), 331

PS2104\_MAX\_TIMEBASE PS3000\_CALLBACK\_FUNC (in module  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.callbacks),  
attribute), 329 221

PS2105\_MAX\_ETS\_CYCLES PS3000A\_ThresholdMode (class in  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.enums),  
attribute), 329 255

PS2105\_MAX\_ETS\_INTERLEAVE PS3000ABandwidthLimiter (class in  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.enums),  
attribute), 329 247

PS2105\_MAX\_TIMEBASE ps3000aBlockReady (in module  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.callbacks),  
attribute), 329 220

PS2200\_MAX\_TIMEBASE PS3000AChannel (class in  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.enums),  
attribute), 329 248

PS2203\_MAX\_ETS\_CYCLES PS3000AChannelBufferIndex (class in  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.enums),  
attribute), 329 248

PS2203\_MAX\_ETS\_INTERLEAVE PS3000AChannelInfo (class in  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.enums),  
attribute), 330 251

PS2204\_MAX\_ETS\_CYCLES PS3000ACoupling (class in  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.enums),  
attribute), 330 251

PS2204\_MAX\_ETS\_INTERLEAVE ps3000aDataReady (in module  
(msl.equipment.resources.picotech.picoscope.ps2000.PicoScope2000resources.picotech.picoscope.callbacks),  
attribute), 330



|  |     |
|--|-----|
| 221  | 253 |
| PS3000ADigitalChannel (class in PS3000AThresholdDirection (class in msl.equipment.resources.picotech.picoscope.enums),         | 249 |
| 249  | 256 |
| PS3000ADigitalChannelDirections (class in PS3000ATimeUnits (class in msl.equipment.resources.picotech.picoscope.structs),      | 350 |
| 350  | 252 |
| PS3000ADigitalDirection (class in PS3000ATriggerChannelProperties (class in msl.equipment.resources.picotech.picoscope.enums), | 256 |
| 256  | 350 |
| PS3000ADigitalPort (class in PS3000ATriggerConditions (class in msl.equipment.resources.picotech.picoscope.enums),             | 249 |
| 249  | 348 |
| PS3000AEtsMode (class in PS3000ATriggerConditionsV2 (class in msl.equipment.resources.picotech.picoscope.enums),               | 252 |
| 252  | 349 |
| PS3000AExtraOperations (class in PS3000ATriggerInfo (class in msl.equipment.resources.picotech.picoscope.enums),               | 254 |
| 254  | 351 |
| PS3000AHoldOffType (class in PS3000ATriggerState (class in msl.equipment.resources.picotech.picoscope.enums),                  | 258 |
| 258  | 257 |
| PS3000AIndexMode (class in PS3000AWaveType (class in msl.equipment.resources.picotech.picoscope.enums),                        | 255 |
| 255  | 253 |
| PS3000APulseWidthType (class in PS3000Channel (class in msl.equipment.resources.picotech.picoscope.enums),                     | 258 |
| 258  | 242 |
| PS3000APwqConditions (class in PS3000Error (class in msl.equipment.resources.picotech.picoscope.structs),                      | 349 |
| 349  | 219 |
| PS3000APwqConditionsV2 (class in PS3000EtsMode (class in msl.equipment.resources.picotech.picoscope.structs),                  | 350 |
| 350  | 245 |
| PS3000ARange (class in PS3000Info (class in msl.equipment.resources.picotech.picoscope.enums),                                 | 250 |
| 250  | 244 |
| PS3000ARatioMode (class in PS3000OpenProgress (class in msl.equipment.resources.picotech.picoscope.enums),                     | 257 |
| 257  | 245 |
| PS3000ASigGenTrigSource (class in PS3000PulseWidthType (class in msl.equipment.resources.picotech.picoscope.enums),            | 254 |
| 254  | 247 |
| PS3000ASigGenTrigType (class in PS3000PwqConditions (class in msl.equipment.resources.picotech.picoscope.enums),               | 254 |
| 254  | 348 |
| ps3000aStreamingReady (in module PS3000Range (class in msl.equipment.resources.picotech.picoscope.callbacks),                  | 221 |
| 221  | 242 |
| PS3000ASweepType (class in PS3000ThresholdDirection (class in msl.equipment.resources.picotech.picoscope.enums),               |     |

246 attribute), 332  
 PS3000ThresholdMode (class in PS3205A\_MAX\_ETS\_CYCLES  
 msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
 246 attribute), 334  
 PS3000TimeUnits (class in PS3205A\_MAX\_INTERLEAVE  
 msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
 243 attribute), 334  
 PS3000TriggerChannelProperties (class in PS3205MSO\_MAX\_INTERLEAVE  
 msl.equipment.resources.picotech.picoscope.structs),(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
 347 attribute), 334  
 PS3000TriggerConditions (class in PS3206\_MAX\_ETS\_CYCLES  
 msl.equipment.resources.picotech.picoscope.structs),(msl.equipment.resources.picotech.picoscope.ps3000.Pico  
 347 attribute), 333  
 PS3000TriggerDirection (class in PS3206\_MAX\_ETS\_INTERLEAVE  
 msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps3000.Pico  
 244 attribute), 333  
 PS3000TriggerState (class in PS3206\_MAX\_TIMEBASE  
 msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps3000.Pico  
 246 attribute), 332  
 PS3000WaveTypes (class in PS3206A\_MAX\_ETS\_CYCLES  
 msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps3000a.Pico  
 243 attribute), 334  
 PS300\_MAX\_ETS\_SAMPLES PS3206A\_MAX\_INTERLEAVE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000a.Pico  
 attribute), 333 attribute), 334  
 PS3204\_MAX\_ETS\_CYCLES PS3206B\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000a.Pico  
 attribute), 333 attribute), 335  
 PS3204\_MAX\_ETS\_INTERLEAVE PS3206MSO\_MAX\_INTERLEAVE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000a.Pico  
 attribute), 333 attribute), 334  
 PS3204\_MAX\_TIMEBASE PS3207A\_MAX\_ETS\_CYCLES  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000a.Pico  
 attribute), 332 attribute), 334  
 PS3204A\_MAX\_ETS\_CYCLES PS3207A\_MAX\_INTERLEAVE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000a.Pico  
 attribute), 334 attribute), 334  
 PS3204A\_MAX\_INTERLEAVE PS3207B\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000a.Pico  
 attribute), 334 attribute), 334  
 PS3204MSO\_MAX\_INTERLEAVE PS3223\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000.Pico  
 attribute), 334 attribute), 333  
 PS3205\_MAX\_ETS\_CYCLES PS3224\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000.Pico  
 attribute), 333 attribute), 333  
 PS3205\_MAX\_ETS\_INTERLEAVE PS3225\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000.Pico  
 attribute), 333 attribute), 333  
 PS3205\_MAX\_TIMEBASE PS3226\_MAX\_TIMEBASE  
 (msl.equipment.resources.picotech.picoscope.ps3000(PicoScope3000a resources.picotech.picoscope.ps3000.Pico

attribute), 333  
 PS3423\_MAX\_TIMEBASE (class in msl.equipment.resources.picotech.picoscope.ps3000aPicoScopeResources), 273  
 PS3424\_MAX\_TIMEBASE (class in msl.equipment.resources.picotech.picoscope.ps3000aPicoScopeResources), 269  
 PS3425\_MAX\_TIMEBASE (class in msl.equipment.resources.picotech.picoscope.ps3000aPicoScopeResources), 279  
 PS3426\_MAX\_TIMEBASE (class in msl.equipment.resources.picotech.picoscope.ps3000aPicoScopeResources), 277  
 PS4000ABandwidthLimiter (class in msl.equipment.resources.picotech.picoscope.enums), 269  
 ps4000aBlockReady (in module PS4000AMetaOperation (class in msl.equipment.resources.picotech.picoscope.enums), 220  
 PS4000AChannel (class in PS4000AMetaType (class in msl.equipment.resources.picotech.picoscope.enums), 270  
 PS4000AChannelBufferIndex (class in PS4000APicoStringValue (class in msl.equipment.resources.picotech.picoscope.enums), 271  
 PS4000AChannelInfo (class in PS4000APulseWidthType (class in msl.equipment.resources.picotech.picoscope.enums), 281  
 PS4000AChannelLed (class in PS4000ARange (class in msl.equipment.resources.picotech.picoscope.enums), 275  
 PS4000AChannelLedSetting (class in PS4000ARatioMode (class in msl.equipment.resources.picotech.picoscope.enums), 352  
 PS4000ACondition (class in PS4000AResistanceRange (class in msl.equipment.resources.picotech.picoscope.enums), 353  
 PS4000AConditionsInfo (class in PS4000ASensorState (class in msl.equipment.resources.picotech.picoscope.enums), 280  
 PS4000AConnectDetect (class in PS4000ASigGenTrigSource (class in msl.equipment.resources.picotech.picoscope.enums), 353  
 PS4000ACoupling (class in PS4000ASigGenTrigType (class in msl.equipment.resources.picotech.picoscope.enums), 270  
 ps4000aDataReady (in module ps4000aStreamingReady (in module msl.equipment.resources.picotech.picoscope.callbacks), 221  
 PS4000ADirection (class in PS4000ASweepType (class in msl.equipment.resources.picotech.picoscope.enums),

|                                 |  |   |     |
|---------------------------------|--|---|-----|
| 274                             |  | 268   |     |
| PS4000AThresholdDirection       | (class in PS4000PulseWidthType                                 | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 278                             |  | 268   |     |
| PS4000AThresholdMode            | (class in PS4000PwqConditions                                  | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.structs</i> ),    |     |
| 277                             |  | 351   |     |
| PS4000ATimeUnits                | (class in PS4000Range  | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 273                             |  | 260   |     |
| PS4000ATriggerChannelProperties | (class in PS4000RatioMode                                      | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.structs</i> ),   | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 353                             |  | 267   |     |
| PS4000ATriggerState             | (class in PS4000SigGenTrigSource                               | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 278                             |  | 265   |     |
| PS4000AWaveType                 | (class in PS4000SigGenTrigType                                 | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 274                             |  | 265   |     |
| ps4000BlockReady                | (in module ps4000StreamingReady                                | (in module  |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.callbacks</i> ), | <i>msl.equipment.resources.picotech.picoscope.callbacks</i> ),  |     |
| 220                             |  | 221   |     |
| PS4000Channel                   | (class in PS4000SweepType                                      | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 259                             |  | 263   |     |
| PS4000ChannelBufferIndex        | (class in PS4000ThresholdDirection                             | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 259                             |  | 266   |     |
| PS4000ChannelInfo               | (class in PS4000ThresholdMode                                  | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 262                             |  | 266   |     |
| ps4000DataReady                 | (in module PS4000TimeUnits                                     | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.callbacks</i> ), | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 221                             |  | 263   |     |
| PS4000EtsMode                   | (class in PS4000TriggerChannelProperties                       | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.structs</i> ),    |     |
| 262                             |  | 352   |     |
| PS4000FrequencyCounterRange     | (class in PS4000TriggerConditions                              | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.structs</i> ),    |     |
| 269                             |  | 351   |     |
| PS4000IndexMode                 | (class in PS4000TriggerState                                   | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 265                             |  | 267   |     |
| PS4000OperationTypes            | (class in PS4000WaveType                                       | (class in   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),      |     |
| 264                             |  | 264   |     |
| PS4000Probe                     | (class in PS4262_MAX_VALUE                                     |   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | ( <i>msl.equipment.resources.picotech.picoscope.ps4000.Pico</i> |     |
| 261                             |  | attribute),   | 336 |
| PS4000Ps4000HoldOffType         | (class in PS4262_MAX_WAVEFORM_BUFFER_SIZE                      |   |     |
|                                 | <i>msl.equipment.resources.picotech.picoscope.enums</i> ),     | ( <i>msl.equipment.resources.picotech.picoscope.ps4000.Pico</i> |     |

|   |   |
|---|---|
| <a href="#">attribute</a> ), 337  | <a href="#">PS5000ARange</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 295                      |
| <a href="#">PS4262_MIN_DWELL_COUNT</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 337    | <a href="#">PS5000ARatioMode</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 300                  |
| <a href="#">PS4262_MIN_VALUE</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 336          | <a href="#">PS5000ASigGenTrigSource</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 297           |
| <a href="#">PS4XXX_MAX_ETS_CYCLES</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 336     | <a href="#">PS5000ASigGenTrigType</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 297             |
| <a href="#">PS4XXX_MAX_INTERLEAVE</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 336     | <a href="#">PS5000ABandwidthLimiter</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 293           |
| <a href="#">PS5000ABandwidthLimiter</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 293   | <a href="#">ps5000aStreamingReady</a> (in module <a href="#">msl.equipment.resources.picotech.picoscope.callbacks</a> ), 221        |
| <a href="#">ps5000aBlockReady</a> (in module <a href="#">msl.equipment.resources.picotech.picoscope.callbacks</a> ), 221    | <a href="#">PS5000ASweepType</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 296                  |
| <a href="#">PS5000AChannel</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 293            | <a href="#">PS5000AThresholdDirection</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 299         |
| <a href="#">PS5000AChannelBufferIndex</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 294 | <a href="#">PS5000AThresholdMode</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 298              |
| <a href="#">PS5000AChannelInfo</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 301        | <a href="#">PS5000ATimeUnits</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 296                  |
| <a href="#">PS5000ACoupling</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 293           | <a href="#">PS5000ATriggerChannelProperties</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.structs</a> ), 356 |
| <a href="#">ps5000aDataReady</a> (in module <a href="#">msl.equipment.resources.picotech.picoscope.callbacks</a> ), 221     | <a href="#">PS5000ATriggerConditions</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.structs</a> ), 355        |
| <a href="#">PS5000ADeviceResolution</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 292   | <a href="#">PS5000ATriggerInfo</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.structs</a> ), 354              |
| <a href="#">PS5000AEtsMode</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 295            | <a href="#">PS5000ATriggerState</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 299               |
| <a href="#">PS5000AExtraOperations</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 292    | <a href="#">PS5000ATriggerWithinPreTrigger</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 300    |
| <a href="#">PS5000AIndexMode</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 298          | <a href="#">PS5000AWaveType</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 296                   |
| <a href="#">PS5000APulseWidthType</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 301     | <a href="#">ps5000BlockReady</a> (in module <a href="#">msl.equipment.resources.picotech.picoscope.callbacks</a> ), 220             |
| <a href="#">PS5000APwqConditions</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.structs</a> ), 301    | <a href="#">PS5000Channel</a> (class in <a href="#">msl.equipment.resources.picotech.picoscope.enums</a> ), 293                     |

285 354  
PS5000ChannelBufferIndex (class in PS5000TriggerConditions (class in  
msl.equipment.resources.picotech.picoscope.enums),msl.equipment.resources.picotech.picoscope.structs),  
285 353  
PS5000ChannelInfo (class in PS5000TriggerState (class in  
msl.equipment.resources.picotech.picoscope.enums),msl.equipment.resources.picotech.picoscope.enums),  
292 290  
ps5000DataReady (in module PS5000WaveType (class in  
msl.equipment.resources.picotech.picoscope.callbacks),msl.equipment.resources.picotech.picoscope.enums),  
221 288  
PS5000EtsMode (class in PS5242A\_MAX\_ETS\_CYCLES  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
286 attribute), 342  
PS5000IndexMode (class in PS5242A\_MAX\_ETS\_INTERLEAVE  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
289 attribute), 342  
PS5000PulseWidthType (class in PS5243A\_MAX\_ETS\_CYCLES  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
291 attribute), 342  
PS5000PwqConditions (class in PS5243A\_MAX\_ETS\_INTERLEAVE  
msl.equipment.resources.picotech.picoscope.structs),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
354 attribute), 342  
PS5000Range (class in PS5244A\_MAX\_ETS\_CYCLES  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
286 attribute), 342  
PS5000RatioMode (class in PS5244A\_MAX\_ETS\_INTERLEAVE  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
291 attribute), 342  
PS5000SigGenTrigSource (class in PS5X42A\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
289 attribute), 341  
PS5000SigGenTrigType (class in PS5X43A\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
msl.equipment.resources.picotech.picoscope.enums),(msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
288 attribute), 341  
ps5000StreamingReady (in module PS5X44A\_MAX\_SIG\_GEN\_BUFFER\_SIZE  
msl.equipment.resources.picotech.picoscope.callbacks),msl.equipment.resources.picotech.picoscope.ps5000a.Pic  
221 attribute), 342  
PS5000SweepType (class in PS6000BandwidthLimiter (class in  
msl.equipment.resources.picotech.picoscope.enums),msl.equipment.resources.picotech.picoscope.enums),  
287 302  
PS5000ThresholdDirection (class in ps6000BlockReady (in module  
msl.equipment.resources.picotech.picoscope.enums),msl.equipment.resources.picotech.picoscope.callbacks),  
290 221  
PS5000ThresholdMode (class in PS6000Channel (class in  
msl.equipment.resources.picotech.picoscope.enums),msl.equipment.resources.picotech.picoscope.enums),  
289 302  
PS5000TimeUnits (class in PS6000ChannelBufferIndex (class in  
msl.equipment.resources.picotech.picoscope.enums),msl.equipment.resources.picotech.picoscope.enums),  
287 303  
PS5000TriggerChannelProperties (class in PS6000Coupling (class in  
msl.equipment.resources.picotech.picoscope.structs),msl.equipment.resources.picotech.picoscope.enums),



|   |  |
|---|--|
| 304   | 357  |
| ps6000DataReady (in module PS6000)                                  | TriggerConditions (class in msl.equipment.resources.picotech.picoscope.callbacks), 221                                   |
| PS6000EtsMode (class in PS6000)                                     | TriggerState (class in msl.equipment.resources.picotech.picoscope.enums), 304  |
| PS6000ExternalFrequency (class in PS6000)                           | WaveType (class in msl.equipment.resources.picotech.picoscope.enums), 301  |
| PS6000ExtraOperations (class in PS640X_C_D_MAX_SIG_GEN_BUFFER_SIZE) | msl.equipment.resources.picotech.picoscope.ps6000.Pico attribute), 343   |
| PS6000IndexMode (class in PT100)                                    | (msl.equipment.resources.picotech.pt104.Pt104DataType msl.equipment.resources.picotech.picoscope.enums), attribute), 358 |
| 307   | PT1000 (msl.equipment.resources.picotech.pt104.Pt104DataType attribute), 358   |
| PS6000PulseWidthType (class in                                      | PT104), (class in msl.equipment.resources.picotech.pt104), 310   |
| PS6000PwqConditions (class in                                       | PT1040DataTypes (class in msl.equipment.resources.picotech.pt104), 357   |
| PS6000Range (class in   | PULSE_WIDTH_ARRAY_OF_BLOCK_CONDITIONS (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283             |
| PS6000RatioMode (class in   | PULSE_WIDTH_CONDITIONS (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                            |
| PS6000SigGenTrigSource (class in                                    | PULSE_WIDTH_CONDITIONS_SOURCE (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                     |
| PS6000SigGenTrigType (class in                                      | PULSE_WIDTH_CONDITIONS_STATE (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                      |
| ps6000StreamingReady (in module                                     | PULSE_WIDTH_NO_OF_BLOCK_CONDITIONS (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                |
| 221   | PULSE_WIDTH_NO_OF_CONDITIONS (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                      |
| PS6000SweepType (class in   | PULSE_WIDTH_PROPERTIES (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                            |
| PS6000ThresholdDirection (class in                                  | PULSE_WIDTH_PROPERTIES_DIRECTION (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                  |
| PS6000ThresholdMode (class in                                       | PULSE_WIDTH_PROPERTIES_LOWER (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                      |
| PS6000TimeUnits (class in   | PULSE_WIDTH_PROPERTIES_TYPE (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                       |
| PS6000TriggerChannelProperties (class in                            | PULSE_WIDTH_PROPERTIES_TYPE (msl.equipment.resources.picotech.picoscope.enums.PS40 attribute), 283                       |
| msl.equipment.resources.picotech.picoscope.enums.PS40               |  |

|                                     |  |  |
|-------------------------------------|--|--|
|                                     | ( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerConditions</i> attribute), 283     | <i>PZ_ControlModeTypes</i> (class in <i>msl.equipment.resources.thorlabs.kinesis.enums</i> ), 433                            |
| <i>PULSE_WIDTH_PROPERTIES_UPPER</i> | ( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerConditions</i> attribute), 283     |  |
| <i>PULSE_WIDTH_SOURCE</i>           | ( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerConditions</i> attribute), 271     | <i>PZ_ControlModeUndefined</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackModeTypes</i> attribute), 450 |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerConditions</i> attribute), 346   | <i>PZ_ExternalSignal</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_InputSourceFlags</i> attribute), 432         |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerConditions</i> attribute), 345   | <i>PZ_FeedbackLoopConstants</i> (class in <i>msl.equipment.resources.thorlabs.kinesis.structs</i> ), 563                     |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerConditions</i> attribute), 349   | <i>PZ_Fixed</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_OutputTrigSenseHigh</i> attribute), 433               |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerConditionsV2</i> attribute), 349 | <i>PZ_InputSourceFlags</i> (class in <i>msl.equipment.resources.thorlabs.kinesis.enums</i> ), 433                            |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerConditions</i> attribute), 348   | <i>PZ_InputTrigEnable</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_OutputTrigSenseHigh</i> attribute), 433     |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerConditions</i> attribute), 351   | <i>PZ_InputTrigSenseHigh</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_OutputTrigSenseHigh</i> attribute), 433  |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerConditions</i> attribute), 355   | <i>PZ_LUTWaveParameters</i> (class in <i>msl.equipment.resources.thorlabs.kinesis.structs</i> ), 433                         |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerConditions</i> attribute), 354   | <i>PZ_OpenLoop</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackModeTypes</i> attribute), 450             |
| <i>pulseWidthQualifier</i>          | ( <i>msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerConditions</i> attribute), 356   | <i>PZ_OpenLoopSmooth</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackModeTypes</i> attribute), 450       |
| <i>PyVISA</i>                       | ( <i>msl.equipment.constants.Backend</i> attribute), 44  | <i>PZ_OpenLoopSmooth</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes</i> attribute), 432         |
| <i>PyVISA_LIBRARY</i>               | ( <i>msl.equipment.config.Config</i> attribute), 18  | <i>PZ_OutputGated</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes</i> attribute), 433            |
| <i>PZ_All</i>                       | ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_InputSourceFlags</i> attribute), 433            | <i>PZ_OutputGated</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes</i> attribute), 433            |
| <i>PZ_CloseLoop</i>                 | ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_OutputTrigSenseHigh</i> attribute), 450         | <i>PZ_OutputTrigRepeat</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes</i> attribute), 432       |
| <i>PZ_CloseLoop</i>                 | ( <i>msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackModeTypes</i> attribute), 450          | <i>PZ_OutputTrigSenseHigh</i> ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes</i> attribute), 432    |
| <i>PZ_CloseLoopSmooth</i>           | ( <i>msl.equipment.resources.thorlabs.kinesis.enums.KNA_FeedbackModeTypes</i> attribute), 450          |  |
| <i>PZ_CloseLoopSmooth</i>           | ( <i>msl.equipment.resources.thorlabs.kinesis.enums.PZ_ControlModeTypes</i> attribute), 432            |  |



(*msl.equipment.resources.thorlabs.kinesis.enums.PZ\_401* attribute), 433

PZ\_Potentiometer (*msl.equipment.resources.thorlabs.kinesis.enums.PZ\_401* attribute), 433

PZ\_SoftwareOnly (*msl.equipment.resources.thorlabs.kinesis.enums.PZ\_401* attribute), 432

PZ\_Undefined (*msl.equipment.resources.thorlabs.kinesis.enums.PZ\_401* attribute), 432

Q

QD\_AutoOpenClosedLoop (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_ClosedLoop (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_Disabled (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 454

QD\_Enabled (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 454

QD\_FilterEnable (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 454

QD\_HoldOnLastValue (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 454

QD\_HoldOnZero (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 454

QD\_HubAndSMA (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_KPA\_DigitalIO (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 575

QD\_KPA\_TrigIOConfig (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 575

QD\_KPA\_TrigModes (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 454

QD\_KPA\_TrigPolarities (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 454

QD\_LoopParameters (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 573

QD\_LowPassFilterParameters (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 573

QD\_LowVoltageRoute (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 255

QD\_ModeUndefined (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_Monitor (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_NcPZ\_TripParameters (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 574

QD\_OpenLoop (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_OpenLoopHoldValues (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 574

QD\_OperatingMode (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_PDParseFilterEnable (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 574

QD\_Position (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 574

QD\_PositionDemandParameters (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 574

QD\_Readings (class in *msl.equipment.resources.thorlabs.kinesis.enums*), 574

QD\_RouteUnQD\_FixedVoltageRoute (class in *msl.equipment.resources.thorlabs.kinesis.enums.QD\_Low* attribute), 453

QD\_SMAOnly (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 453

QD\_Trig\_Disabled (class in *msl.equipment.resources.thorlabs.kinesis.enums.QD\_KPA* attribute), 454

QD\_TrigIn\_GPI (class in *msl.equipment.resources.thorlabs.kinesis.enums.QD\_KPA* attribute), 454

QD\_TrigIn\_LoopOpenClose (class in *msl.equipment.resources.thorlabs.kinesis.enums.QD\_KPA* attribute), 454

QD\_Undefined (*msl.equipment.resources.thorlabs.kinesis.enums.QD\_401* attribute), 454

QUAD (in module *msl.equipment.resources.thorlabs.kinesis.messages*), 550

QUAD (*msl.equipment.resources.picotech.picoscope.enums.PS2000A* attribute), 239

QUAD (*msl.equipment.resources.picotech.picoscope.enums.PS3000A* attribute), 255

QUAD (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Mode  
 attribute), 277 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS3000AIndex.Mode  
 attribute), 266 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Mode  
 attribute), 298 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Mode  
 attribute), 289 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndex.Mode  
 attribute), 308 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndex.Mode  
 attribute), 294 query() (msl.equipment.connection\_message\_based.ConnectionMessageBased  
 method), 25 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS6000AIndex.Mode  
 attribute), 303 query() (msl.equipment.connection\_prologix.ConnectionPrologix  
 method), 29 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Mode  
 attribute), 234 query\_auto (msl.equipment.connection\_prologix.ConnectionPrologix  
 property), 29 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Mode  
 attribute), 225 Quickest (msl.equipment.resources.thorlabs.kinesis.enums.MOFiberMode.Directions  
 attribute), 424 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS3000AIndex.Mode  
 attribute), 251

## R

R\_100MV (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Range  
 attribute), 234 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Range  
 attribute), 225 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Range  
 attribute), 251 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndex.Range  
 attribute), 243 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndex.Range  
 attribute), 272 R\_10V (msl.equipment.resources.picotech.picoscope.enums.PS6000AIndex.Range  
 attribute), 260 R\_1100K (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Range  
 attribute), 295 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Range  
 attribute), 286 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Range  
 attribute), 303 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS3000AIndex.Range  
 attribute), 243 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS3000AIndex.Range  
 attribute), 272 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Range  
 attribute), 260 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS4000AIndex.Range  
 attribute), 273 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Range  
 attribute), 234 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS5000AIndex.Range  
 attribute), 225 R\_10MV (msl.equipment.resources.picotech.picoscope.enums.PS2000AIndex.Range  
 attribute), 225 R\_1V (msl.equipment.resources.picotech.picoscope.enums.PS6000AIndex.Range  
 attribute), 234

[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS2000A\\_range\\_attribute\), 234](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS2000A\\_range\\_attribute\), 225](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 251](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 243](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 272](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 260](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 295](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 286](#)  
[R\\_200MV \(msl.equipment.resources.picotech.picoscope.enums.PS6000A\\_range\\_attribute\), 304](#)  
[R\\_200V \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 243](#)  
[R\\_200V \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 272](#)  
[R\\_200V \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 295](#)  
[R\\_200V \(msl.equipment.resources.picotech.picoscope.enums.PS6000A\\_range\\_attribute\), 304](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS2000A\\_range\\_attribute\), 234](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS2000A\\_range\\_attribute\), 225](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 251](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 243](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 272](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 260](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 295](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 286](#)  
[R\\_20MV \(msl.equipment.resources.picotech.picoscope.enums.PS6000A\\_range\\_attribute\), 304](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS2000A\\_range\\_attribute\), 234](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS2000A\\_range\\_attribute\), 225](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 251](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS3000A\\_range\\_attribute\), 243](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 272](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 260](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 295](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS5000A\\_range\\_attribute\), 286](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS6000A\\_range\\_attribute\), 304](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS6000A\\_range\\_attribute\), 272](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 260](#)  
[R\\_20V \(msl.equipment.resources.picotech.picoscope.enums.PS4000A\\_range\\_attribute\), 225](#)





attribute), 335

RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.enums.RS4000A.WaveType attribute), 339

RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000A.ProScope5000 attribute), 341

RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps5000A.ProScope5000 attribute), 342

RAMP\_MAX\_FREQUENCY (msl.equipment.resources.picotech.picoscope.ps6000A.PithusPro6000 attribute), 344

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS2000A.WaveType attribute), 237

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS3000A.WaveType attribute), 253

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS4000A.WaveType attribute), 274

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS4000A.WaveType attribute), 264

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS5000A.WaveType attribute), 297

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS5000A.WaveType attribute), 288

RAMP\_UP (msl.equipment.resources.picotech.picoscope.enums.RS6000A.WaveType attribute), 306

RAMPDOWN (msl.equipment.resources.picotech.picoscope.enums.RS2000A.WaveType attribute), 228

RAMPUP (msl.equipment.resources.picotech.picoscope.enums.RS2000A.WaveType attribute), 228

RANGES (msl.equipment.resources.picotech.picoscope.enums.RS2000A.ChannelInfo attribute), 235

RANGES (msl.equipment.resources.picotech.picoscope.enums.RS3000A.ChannelInfo attribute), 252

RANGES (msl.equipment.resources.picotech.picoscope.enums.RS4000A.ChannelInfo attribute), 281

RANGES (msl.equipment.resources.picotech.picoscope.enums.RS4000A.ChannelInfo attribute), 262

RANGES (msl.equipment.resources.picotech.picoscope.enums.RS5000A.ChannelInfo attribute), 301

RANGES (msl.equipment.resources.picotech.picoscope.enums.RS5000A.ChannelInfo attribute), 292

raw (msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel property), 223

READ (msl.equipment.resources.picotech.picoscope.enums.RS4000A.Autoparallel attribute), 276

read() (msl.equipment.connection\_message\_based.RegBusio(msl.equipment.resources.nkt.nktpdll.RegisterStatusTypes attribute), 24

read() (msl.equipment.connection\_prologix.ConnectionPrologix.RegConfig attribute), 29

read() (msl.equipment.hislip.HiSLIPClient method), 67

read() (msl.equipment.resources.bentham.benhw32.Bentham32 method), 123

read() (msl.equipment.vxi11.RPCClient method),

read\_authentication\_exchange() (msl.equipment.hislip.SyncClient method), 70

read\_ol756\_flash\_settings() (msl.equipment.resources.optronic\_laboratories.ol756ocx attribute), 26000

read\_reply() (msl.equipment.vxi11.VXIClient method), 320

read\_stb() (msl.equipment.connection\_tcpip\_hislip.ConnectionTCP attribute), 28

read\_stb() (msl.equipment.connection\_tcpip\_vxi11.ConnectionTCP attribute), 28

read\_termination (msl.equipment.connection\_message\_based.ConnectionMessage property), 23

read\_wave\_type (msl.equipment.connection\_prologix.ConnectionPrologix attribute), 28

recall() (msl.equipment.resources.aim\_tti.mx\_series.MXSeries attribute), 306

recall\_all() (msl.equipment.resources.aim\_tti.mx\_series.MXSeries attribute), 306

receive() (msl.equipment.hislip.SyncClient method), 67

reconnect() (msl.equipment.connection\_socket.ConnectionSocket attribute), 36

reconnect() (msl.equipment.connection\_tcpip\_hislip.ConnectionTCP attribute), 36

reconnect() (msl.equipment.connection\_tcpip\_vxi11.ConnectionTCP attribute), 36

reconnect() (msl.equipment.connection\_zeromq.ConnectionZero attribute), 36

Record (class in msl.equipment.record\_types), 80

Record (class in msl.equipment.record\_types), 80

records (msl.equipment.database.Database method), 46

RED (msl.equipment.resources.picotech.picoscope.enums.PS4000A.ChannelInfo attribute), 275

Register (msl.equipment.resources.nkt.nktpdll.NKT.RegisterStatusTypes attribute), 163

Register (msl.equipment.resources.nkt.nktpdll.NKT.RegisterStatusTypes attribute), 158

Register (msl.equipment.resources.nkt.nktpdll.NKT.RegisterStatusTypes attribute), 163

RegComError (msl.equipment.resources.nkt.nktpdll.RegisterStatusType), 158  
attribute), 158 RegData\_H8 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegCRCErr (msl.equipment.resources.nkt.nktpdll.NKT.RegisterStatusType), 162  
attribute), 163 RegData\_H8 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegCRCErr (msl.equipment.resources.nkt.nktpdll.RegisterStatusType), 157  
attribute), 158 RegData\_Mixed (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_Ascii (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_Mixed (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_Ascii (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157  
attribute), 157 RegData\_Paramset (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162

RegData\_B16 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_Paramset (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_B16 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157  
attribute), 157 RegData\_S16 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_B32 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_S16 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157

RegData\_B32 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162  
attribute), 158 RegData\_S16 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162

RegData\_B64 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_S32 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157

RegData\_B64 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162  
attribute), 158 RegData\_S32 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162

RegData\_B8 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_S64 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157

RegData\_B8 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162  
attribute), 157 RegData\_S64 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 162

RegData\_DateTime (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_S8 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_DateTime (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157  
attribute), 158 RegData\_U16 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_F32 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_U16 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_F32 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157  
attribute), 157 RegData\_U32 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_F64 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_U32 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_F64 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157  
attribute), 157 RegData\_U64 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_H16 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_U64 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_H16 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157  
attribute), 157 RegData\_U8 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_H32 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_U8 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157

RegData\_H32 (msl.equipment.resources.nkt.nktpdll.RegisterDataTypes), 157  
attribute), 158 RegData\_Unknown (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162

RegData\_H64 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 162  
attribute), 162 RegData\_Unknown (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157

RegData\_H64 (msl.equipment.resources.nkt.nktpdll.NKT.RegisterDataTypes), 157  
attribute), 158

|   |  |
|---|--|
| <code>(msl.equipment.resources.nkt.nktpdll.RegisterDisplacementRead_s64()</code><br><code>attribute), 157</code>  | <code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>register_read_s64()</code>  |
| <code>register()</code> (in module <code>msl.equipment.resources</code> ), 81   | <code>register_read_s8()</code>  |
| <code>register()</code> ( <code>msl.equipment.resources.avantes.avaspec.Avantec</code><br><code>method</code> ), 117                                      | <code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 173  |
| <code>register_create()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 171                                     | <code>register_read_u16()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 174                      |
| <code>register_exists()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 171                                     | <code>register_read_u32()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 174                      |
| <code>register_get_all()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 171                                    | <code>register_read_u64()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 174                      |
| <code>register_message_callback()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.benchmark_top_staple</code><br><code>method</code> ), 404     | <code>register_read_u8()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.benchmark_top_staple</code><br><code>method</code> ), 175 |
| <code>register_message_callback()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.filter_flipped</code><br><code>method</code> ), 467           | <code>register_remove()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 175                        |
| <code>register_message_callback()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motor</code><br><code>method</code> ), 479 | <code>register_remove_all()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 175                    |
| <code>register_message_callback()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code><br><code>method</code> ), 502           | <code>register_write()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 175                         |
| <code>register_message_callback()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide</code><br><code>method</code> ), 519          | <code>register_write_ascii()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 176                   |
| <code>register_message_callback()</code><br><code>(msl.equipment.resources.thorlabs.kinesis.kcube_staple</code><br><code>method</code> ), 536             | <code>register_write_f32()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 176                     |
| <code>register_read()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 170                                       | <code>register_write_f64()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 176                     |
| <code>register_read_ascii()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 171                                 | <code>register_write_read()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 177                    |
| <code>register_read_f32()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 172                                   | <code>register_write_read_ascii()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 177              |
| <code>register_read_f64()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 172                                   | <code>register_write_read_f32()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 177                |
| <code>register_read_s16()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 172                                   | <code>register_write_read_f64()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 178                |
| <code>register_read_s32()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 173                                   | <code>register_write_read_s16()</code><br><code>(msl.equipment.resources.nkt.nktpdll.NKT</code><br><code>method</code> ), 178                |

|  |   |
|--|---|
| <code>register_write_read_s32()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 179 | <code>RegisterStatusCallback</code> (in <i>module</i><br><i>msl.equipment.resources.nkt.nktpdll</i> ),<br>153                   |
| <code>register_write_read_s64()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 179 | <code>RegisterStatusCallback</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>attribute</i> ), 159              |
| <code>register_write_read_s8()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 179  | <code>RegisterStatusTypes</code> (class in<br><i>msl.equipment.resources.nkt.nktpdll</i> ),<br>158                              |
| <code>register_write_read_u16()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 180 | <code>RegNacked</code> ( <i>msl.equipment.resources.nkt.nktpdll.NKT.RegisterStat</i><br><i>attribute</i> ), 163                 |
| <code>register_write_read_u32()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 180 | <code>RegNacked</code> ( <i>msl.equipment.resources.nkt.nktpdll.RegisterStatusTyp</i><br><i>attribute</i> ), 158                |
| <code>register_write_read_u64()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 180 | <code>RegPriority_High</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT.RegisterPriorit</i><br><i>attribute</i> ), 161    |
| <code>register_write_read_u8()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 181  | <code>RegPriority_High</code><br>( <i>msl.equipment.resources.nkt.nktpdll.RegisterPriorityType</i><br><i>attribute</i> ), 157   |
| <code>register_write_s16()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 181      | <code>RegPriority_Low</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT.RegisterPriorit</i><br><i>attribute</i> ), 161     |
| <code>register_write_s32()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 181      | <code>RegPriority_Low</code><br>( <i>msl.equipment.resources.nkt.nktpdll.RegisterPriorityType</i><br><i>attribute</i> ), 157    |
| <code>register_write_s64()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 182      | <code>RegSuccess</code> ( <i>msl.equipment.resources.nkt.nktpdll.NKT.RegisterSta</i><br><i>attribute</i> ), 162                 |
| <code>register_write_s8()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 182       | <code>RegSuccess</code> ( <i>msl.equipment.resources.nkt.nktpdll.RegisterStatusTy</i><br><i>attribute</i> ), 158                |
| <code>register_write_u16()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 182      | <code>RegTimeout</code> ( <i>msl.equipment.resources.nkt.nktpdll.NKT.RegisterSta</i><br><i>attribute</i> ), 163                 |
| <code>register_write_u32()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 183      | <code>RegTimeout</code> ( <i>msl.equipment.resources.nkt.nktpdll.RegisterStatusTy</i><br><i>attribute</i> ), 158                |
| <code>register_write_u64()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 183      | <code>RejectStatus</code> (class in <i>msl.equipment.vxi11</i> ),<br>587  |
| <code>register_write_u8()</code><br>( <i>msl.equipment.resources.nkt.nktpdll.NKT</i><br><i>method</i> ), 183       | <code>relativeReading</code><br>( <i>msl.equipment.resources.thorlabs.kinesis.structs.KNA_TH</i><br><i>attribute</i> ), 570     |
| <code>RegisterDataTypes</code> (class in<br><i>msl.equipment.resources.nkt.nktpdll</i> ),<br>157                   | <code>relativeReading</code><br>( <i>msl.equipment.resources.thorlabs.kinesis.structs.NT_TIA</i><br><i>attribute</i> ), 563     |
| <code>RegisterPriorityTypes</code> (class in<br><i>msl.equipment.resources.nkt.nktpdll</i> ),<br>156               | <code>release_stream_buffer()</code><br>( <i>msl.equipment.resources.picotech.picoscope.ps3000.Pico</i><br><i>method</i> ), 333 |
|  | <code>remote()</code> ( <i>msl.equipment.connection_tcpip_vxi11.ConnectionTCP</i><br><i>method</i> ), 41                        |
|  | <code>remote_local_control()</code><br>( <i>msl.equipment.connection_tcpip_hislip.ConnectionTCP</i><br><i>method</i> ), 38      |
|  | <code>RemoteHostIp</code> ( <i>msl.equipment.resources.avantes.avaspec.Avantes</i><br><i>attribute</i> ), 104                   |



|              |            |
|--------------|------------|
| <b>Index</b> | <b>709</b> |
|--------------|------------|

|  |   |
|--|---|
| <code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 536</code>  | <code>request_move_relative_distance()</code>   |
| <code>request_joystick_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 405</code>           | <code>request_move_relative_distance()</code>   |
| <code>request_key()</code> <code>(misl.equipment.resources.mks_instruments.p4000bpr4000b.request_move_absolute_position()</code><br><code>method), 148</code>                                    | <code>method), 537</code>   |
| <code>request_led_switches()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.request_move_absolute_position()</code><br><code>method), 503</code>                      | <code>request_operating_mode()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_solenoid.request_move_absolute_position()</code><br><code>method), 520</code>                     |
| <code>request_led_switches()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_solenoid.request_move_absolute_position()</code><br><code>method), 519</code>                      | <code>request_operating_state()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_solenoid.request_move_absolute_position()</code><br><code>method), 520</code>                    |
| <code>request_limit_switch_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 405</code>       | <code>request_pid_loop_encoder_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 406</code>    |
| <code>request_limit_switch_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 479</code>    | <code>request_pid_loop_encoder_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 537</code> |
| <code>request_limit_switch_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.request_move_absolute_position()</code><br><code>method), 503</code>               | <code>request_pos_trigger_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.request_move_absolute_position()</code><br><code>method), 503</code>                 |
| <code>request_limit_switch_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 537</code>          | <code>request_pos_trigger_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 537</code>            |
| <code>request_mmi_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.request_move_absolute_position()</code><br><code>method), 503</code>                        | <code>request_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 406</code>                   |
| <code>request_mmi_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_solenoid.request_move_absolute_position()</code><br><code>method), 519</code>                        | <code>request_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 480</code>                |
| <code>request_mmi_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 537</code>                   | <code>request_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.request_move_absolute_position()</code><br><code>method), 503</code>                           |
| <code>request_move_absolute_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 405</code>    | <code>request_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 537</code>                      |
| <code>request_move_absolute_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 480</code> | <code>request_potentiometer_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 480</code>    |
| <code>request_move_absolute_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.request_move_absolute_position()</code><br><code>method), 503</code>            | <code>request_power_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 406</code>               |
| <code>request_move_absolute_position()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 537</code>       | <code>request_power_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 480</code>            |
| <code>request_move_relative_distance()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 405</code>    | <code>request_power_params()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.request_move_absolute_position()</code><br><code>method), 537</code>                  |
| <code>request_move_relative_distance()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.request_move_absolute_position()</code><br><code>method), 480</code> | <code>request_rack_digital_outputs()</code><br><code>(misl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.request_move_absolute_position()</code><br><code>method), 406</code>       |

|   |  |
|---|--|
| <code>method)</code> , 406  | <code>method)</code> , 520   |
| <code>request_rack_status_bits()</code>   | <code>request_trigger_config_params()</code>   |
| <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.benchmarksteppermotors.kinesis.kcube_stepper_motors)</code> , 406             | <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.benchmarksteppermotors.kinesis.kcube_stepper_motors)</code> , 538    |
| <code>request_settings()</code>   | <code>request_trigger_switches()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.benchmarksteppermotors.kinesis.benchtop_stepper_motors)</code> , 406          | <code>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.benchmarksteppermotors.kinesis.benchtop_stepper_motors)</code> , 407 |
| <code>request_settings()</code>   | <code>request_trigger_switches()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.filter_flippartfilterflippart.kinesis.integrated_stepper_motors.integratedsteppermotors)</code> , 467 | <code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.integratedsteppermotors)</code> , 481                              |
| <code>request_settings()</code>   | <code>request_vel_params()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.integratedsteppermotors.kinesis.benchtop_stepper_motors)</code> , 480       | <code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.integratedsteppermotors)</code> , 407                              |
| <code>request_settings()</code>   | <code>request_vel_params()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.kinesis.kcube_dc_servo)</code> , 503   | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.kinesis.kcube_dc_servo)</code> , 481  |
| <code>request_settings()</code>   | <code>request_vel_params()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.kinesis.kcube_solenoide)</code> , 520   | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.kinesis.kcube_dc_servo)</code> , 504  |
| <code>request_settings()</code>   | <code>request_vel_params()</code>  |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.kinesis.kcube_stepper_motors)</code> , 537                                       | <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.kinesis.kcube_stepper_motors)</code> , 538                              |
| <code>request_status()</code>   | <code>RES_12BIT</code> ( <code>msl.equipment.resources.picotech.picoscope.enums.PS_12BIT</code> ), 292                                       |
| <code>(msl.equipment.resources.thorlabs.kinesis.filter_flippartfilterflippart)</code> , 466   | <code>RES_14BIT</code> ( <code>msl.equipment.resources.picotech.picoscope.enums.PS_14BIT</code> ), 292                                       |
| <code>request_status()</code>   | <code>RES_15BIT</code> ( <code>msl.equipment.resources.picotech.picoscope.enums.PS_15BIT</code> ), 292                                       |
| <code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.integratedsteppermotors)</code> , 480                                       | <code>RES_16BIT</code> ( <code>msl.equipment.resources.picotech.picoscope.enums.PS_16BIT</code> ), 292                                       |
| <code>request_status()</code>   | <code>RES_8BIT</code> ( <code>msl.equipment.resources.picotech.picoscope.enums.PS_8BIT</code> ), 292   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.kinesis.kcube_solenoide)</code> , 520   | <code>reserved</code> ( <code>msl.equipment.resources.avantes.avaspec.BroadcastAnalog</code> ), 104  |
| <code>request_status_bits()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KLD_1</code> ), 569   |
| <code>(msl.equipment.resources.thorlabs.kinesis.benchmarksteppermotors.benchmarksteppermotors)</code> , 407   | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KLS_A</code> ), 570   |
| <code>request_status_bits()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMOT_1</code> ), 568  |
| <code>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.integratedsteppermotors)</code> , 480                                       | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMOT_2</code> ), 568  |
| <code>request_status_bits()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KNA_1</code> ), 571   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.kinesis.kcube_dc_servo)</code> , 503   | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KNA_2</code> ), 571   |
| <code>request_status_bits()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_1</code> ), 572   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.kinesis.kcube_solenoide)</code> , 520   | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_2</code> ), 572   |
| <code>request_status_bits()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_3</code> ), 572   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.kinesis.kcube_stepper_motors)</code> , 538                                       | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_4</code> ), 572   |
| <code>request_trigger_config_params()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_5</code> ), 572   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.kinesis.kcube_dc_servo)</code> , 504   | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_6</code> ), 572   |
| <code>request_trigger_config_params()</code>  | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_7</code> ), 572   |
| <code>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.kinesis.kcube_solenoide)</code> , 520   | <code>reserved</code> ( <code>msl.equipment.resources.thorlabs.kinesis.structs.KMZ_8</code> ), 572   |

[attribute](#)), 573  
[reserved](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisMMIP88](#)  
[attribute](#)), 576  
[reserved](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisTb21Config](#)  
[attribute](#)), 576  
[reserved](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisMMIP88](#)  
[attribute](#)), 577  
[reserved](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisTb21Config](#)  
[attribute](#)), 577  
[reserved0](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4000ATTriggerInfo](#)  
[attribute](#)), 351  
[reserved0](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4000ATTriggerInfo](#)  
[attribute](#)), 354  
[reserved1](#) ([msl.equipment.resources.picotech.picoscope.structs.PS1000MTriggerInfo](#)  
[attribute](#)), 351  
[reserved1](#) ([msl.equipment.resources.picotech.picoscope.structs.PS4000ATTriggerInfo](#)  
[attribute](#)), 355  
[reserved1](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisSettings](#)  
[attribute](#)), 566  
[reserved1](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisSettings](#)  
[attribute](#)), 569  
[reserved1](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisTriggerParams](#)  
[attribute](#)), 570  
[reserved1](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisTriggerParams](#)  
[attribute](#)), 557  
[reserved1](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 565  
[reserved2](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 566  
[reserved2](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 569  
[reserved2](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 570  
[reserved2](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reserved3](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reserved4](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reserved5](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reserved6](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reserved7](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reserved8](#) ([msl.equipment.resources.thorlabs.kinesis.structs.KinesisPage\\_IOS\\_defaults](#)  
[attribute](#)), 558  
[reset\(\)](#) ([msl.equipment.resources.aim\\_tti.mx\\_series.MXSeries](#)  
[method](#)), 89  
[reset\(\)](#) ([msl.equipment.resources.energetiq.eq99.EQ99](#)  
[method](#)), 136



[attribute](#)), 261  
[RESISTANCE\\_5K](#) ([msl.equipment.resources.picotech.picoscope.enums.PS3000Attribute](#)), 261  
[RESISTANCE\\_TO\\_10K](#) ([msl.equipment.resources.picotech.pt104.Pt104DataTypes](#) attribute), 254  
[RESISTANCE\\_TO\\_375R](#) ([msl.equipment.resources.picotech.pt104.Pt104DataTypes](#) attribute), 256  
[RESISTANCES](#) ([msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelInfo](#) attribute), 281  
[RESISTANCES](#) ([msl.equipment.resources.picotech.picoscope.enums.PS4000ChannelInfo](#) attribute), 262  
[resource](#) ([msl.equipment.connection\\_pyvisa.ConnectionPyVISA](#) attribute), 278  
[resource](#) property), 31  
[resource\\_class\(\)](#) ([msl.equipment.connection\\_pyvisa.ConnectionPyVISA](#) static method), 31  
[resource\\_manager\(\)](#) ([msl.equipment.connection\\_pyvisa.ConnectionPyVISA](#) static method), 31  
[ResourceClassNotFound](#), 48  
[restPercentage](#) ([msl.equipment.resources.thorlabs.kinesis.structs.MQTTParameters](#) attribute), 561  
[resume\\_move\\_messages\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.b2bapi.B2BAPI](#) method), 407  
[resume\\_move\\_messages\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_cube\\_dc\\_servo](#) method), 504  
[resume\\_move\\_messages\(\)](#) ([msl.equipment.resources.thorlabs.kinesis.kcube\\_dc\\_cube\\_dc\\_servo](#) method), 538  
[Reverse](#) ([msl.equipment.resources.thorlabs.kinesis.enums.ReverseDirection](#) attribute), 439  
[Reverse](#) ([msl.equipment.resources.thorlabs.kinesis.enums.ReverseDirection](#) attribute), 424  
[Reverse](#) ([msl.equipment.resources.thorlabs.kinesis.enums.ReverseDirection](#) attribute), 460  
[rightButtonPosition](#) ([msl.equipment.resources.thorlabs.kinesis.structs.MQTTParameters](#) attribute), 566  
[RISING](#) ([msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection](#) attribute), 240  
[RISING](#) ([msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection](#) attribute), 238  
[RISING](#) ([msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection](#) attribute), 239  
[RISING](#) ([msl.equipment.resources.picotech.picoscope.enums.PS2000ADigitalDirection](#) attribute), 226

|   |  |
|---|--|
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection</i><br>attribute), 257 | <i>run_streaming_ex()</i><br>( <i>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope</i><br>method), 338                              |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection</i><br>attribute), 256 | <i>run_streaming_ns()</i><br>( <i>msl.equipment.resources.picotech.picoscope.picoscope_2</i><br>method), 317                                   |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS3000AThresholdDirection</i><br>attribute), 246 | <i>run_time()</i> ( <i>msl.equipment.resources.energetiq.eq99.EQ99</i><br>method), 317   |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection</i><br>attribute), 278 | <b>S</b><br><i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS2000ATime</i><br>attribute), 236                                  |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000AThresholdDirection</i><br>attribute), 267 | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS2000Time</i><br>attribute), 236   |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection</i><br>attribute), 299 | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS3000ATime</i><br>attribute), 253  |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection</i><br>attribute), 290 | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS3000Time</i><br>attribute), 244   |
| RISING_OR_FALLING<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS5000AThresholdDirection</i><br>attribute), 308 | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000ATime</i><br>attribute), 274  |
| <i>rmt</i> ( <i>msl.equipment.hislip.SyncClient</i> property), 69   | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000Time</i><br>attribute), 263   |
| <i>root</i> ( <i>msl.equipment.config.Config</i> property), 18  | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS5000ATime</i><br>attribute), 296  |
| ROOT_NAME_LEN ( <i>msl.equipment.resources.avantes.avaspec.Avantec</i><br>attribute), 100                                   | <i>S</i> ( <i>msl.equipment.resources.picotech.picoscope.enums.PS6000Time</i><br>attribute), 305   |
| RotationalUnlimited<br>( <i>msl.equipment.resources.thorlabs.kinesis.enums.MOTION_MODES</i><br>attribute), 424              | SAMPLE_PROPERTIES<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000Time</i><br>attribute), 280                                   |
| RotationalWrapping<br>( <i>msl.equipment.resources.thorlabs.kinesis.enums.MOTION_MODES</i><br>attribute), 424               | SAMPLE_PROPERTIES_NO_OF_CAPTURES<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000Time</i><br>attribute), 280                    |
| RPC_MISMATCH ( <i>msl.equipment.vxi11.RejectStatus</i><br>attribute), 587   | SAMPLE_PROPERTIES_OVERLAPPED<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000Time</i><br>attribute), 283                        |
| RPCClient ( <i>class in msl.equipment.vxi11</i> ), 588  | SAMPLE_PROPERTIES_OVERLAPPED_DOWN_SAMPLE_RATIO<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000Time</i><br>attribute), 283      |
| RS232 ( <i>msl.equipment.resources.avantes.avaspec.Avantec.IntegratedType</i><br>attribute), 102                            | SAMPLE_PROPERTIES_OVERLAPPED_DOWN_SAMPLE_RATIO_MODE<br>( <i>msl.equipment.resources.picotech.picoscope.enums.PS4000Time</i><br>attribute), 283 |
| RS232 ( <i>msl.equipment.resources.avantes.avaspec.IntegratedType</i><br>attribute), 92                                     | <i>rstrip</i> ( <i>msl.equipment.connection_message_based.ConnectionMessageBased</i><br>property), 23  |
| <i>rstrip</i> ( <i>msl.equipment.connection_message_based.ConnectionMessageBased</i><br>property), 23                       | <i>rstrip</i> ( <i>msl.equipment.connection_prologix.ConnectionPrologix</i><br>property), 28   |
| <i>rstrip</i> ( <i>msl.equipment.connection_prologix.ConnectionPrologix</i><br>property), 28                                | <i>run_block()</i> ( <i>msl.equipment.resources.picotech.picoscope.picoscope_2</i><br>method), 314   |
| <i>run_block()</i> ( <i>msl.equipment.resources.picotech.picoscope.picoscope_2</i><br>method), 314                          | <i>run_streaming()</i><br>( <i>msl.equipment.resources.picotech.picoscope.picoscope_2</i><br>method), 314                                      |

attribute), 284  
 SAMPLE\_PROPERTIES\_POST\_TRIGGER\_SAMPLES (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoScopeValues), 455  
 attribute), 283  
 SAMPLE\_PROPERTIES\_PRE\_TRIGGER\_SAMPLES SC\_SolenoidClosed (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoScopeValues), 456  
 attribute), 283  
 SAMPLE\_PROPERTIES\_RESOLUTION SC\_SolenoidOpen (msl.equipment.resources.picotech.picoscope.enums.PS4000APicoScopeValues), 456  
 attribute), 283  
 SAMPLE\_PROPERTIES\_TIMEBASE SC\_SolenoidStates (class in msl.equipment.resources.picotech.picoscope.enums.PS4000APicoScopeValues), 456  
 attribute), 283  
 samplesPerRevolution SC\_Triggered (msl.equipment.resources.thorlabs.kinesis.enums.SamplesPerRevolution), 562  
 attribute), 562  
 SAT\_PEAK\_INVERSION SC\_Unknown (msl.equipment.resources.thorlabs.kinesis.enums.SAT\_PEAK\_INVERSION), 455  
 attribute), 455  
 (msl.equipment.resources.avantes.avaspec.Avaspec), 101  
 save() (msl.equipment.resources.aim\_tti.mx\_series.SCOPE\_TRIG), 89  
 attribute), 238  
 save() (msl.equipment.resources.thorlabs.fwx2c.Firmware), 581  
 attribute), 255  
 save\_all() (msl.equipment.resources.aim\_tti.mx\_series.SCOPE\_TRIG), 89  
 attribute), 277  
 save\_calibration\_file() SCOPE\_TRIG (msl.equipment.resources.picotech.picoscope.enums.SCOPE\_TRIG), 210  
 attribute), 210  
 save\_measurement\_data() SCOPE\_TRIG (msl.equipment.resources.picotech.picoscope.enums.SCOPE\_TRIG), 210  
 attribute), 289  
 save\_setup() (msl.equipment.resources.bentham.bioscope.BIOSCOPE2\_TRIG), 123  
 attribute), 307  
 save\_streaming\_data() sdk (msl.equipment.connection\_sdk.ConnectionSDK), 333  
 attribute), 333  
 SC\_Active (msl.equipment.resources.thorlabs.kinesis.enums.SC\_Active), 455  
 attribute), 455  
 SC\_Auto (msl.equipment.resources.thorlabs.kinesis.enums.SC\_Auto), 455  
 attribute), 455  
 SC\_CycleParameters (class in msl.equipment.resources.thorlabs.kinesis.structs), 575  
 attribute), 575  
 SC\_Inactive (msl.equipment.resources.thorlabs.kinesis.enums.SC\_Inactive), 455  
 attribute), 455  
 SC\_Manual (msl.equipment.resources.thorlabs.kinesis.enums.SC\_Manual), 455  
 attribute), 455  
 SC\_OperatingModes (class in msl.equipment.resources.thorlabs.kinesis.enums), 455  
 attribute), 455  
 SC\_OperatingStates (class in segmentIndex (msl.equipment.resources.picotech.picoscope.structs)), 455  
 attribute), 351

attribute), 355  
 select\_lamp() (msl.equipment.resources.optronic\_laboratories.ol755.OpticalSource attribute), 92  
 method), 217 SENS\_HAMS11155\_2048  
 select\_wavelength() (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 method), 123 SENS\_HAMS11155\_2048  
 select\_wavelength() (msl.equipment.resources.avantes.avaspec.SensType attribute), 93  
 method), 125 SENS\_HAMS11638  
 selected\_addresses (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 attribute), 27 SENS\_HAMS11638  
 selectedRange (msl.equipment.resources.thorlabs.kinesis.stk.kn1.TIAReading attribute), 93  
 attribute), 570 SENS\_HAMS11639  
 selectedRange (msl.equipment.resources.thorlabs.kinesis.stk.kn1.TIAReading attribute), 563  
 attribute), 563 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 send() (msl.equipment.hislip.SyncClient method), attribute), 103  
 69 SENS\_HAMS11639  
 send() (msl.equipment.resources.bentham.benhw32.Bentham32 attribute), 93  
 method), 123 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93  
 send\_down\_parameters() SENS\_HAMS12443  
 (msl.equipment.resources.optronic\_laboratories.ol755.OpticalSource attribute), 103  
 method), 210 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 SENS\_HAMG13913 SENS\_HAMS12443  
 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93  
 SENS\_HAMG13913 SENS\_HAMS13496  
 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 SENS\_HAMG9208\_512 SENS\_HAMS13496  
 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93  
 SENS\_HAMG9208\_512 SENS\_HAMS7031\_1024X122  
 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 SENS\_HAMS10420\_2048X64 SENS\_HAMS7031\_1024X122  
 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93  
 SENS\_HAMS10420\_2048X64 SENS\_HAMS7031\_1024X58  
 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 SENS\_HAMS11071\_2048X16 SENS\_HAMS7031\_1024X58  
 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93  
 SENS\_HAMS11071\_2048X16 SENS\_HAMS8378\_1024  
 (msl.equipment.resources.avantes.avaspec.SensType attribute), 93 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103  
 SENS\_HAMS11071\_2048X64 SENS\_HAMS8378\_1024  
 (msl.equipment.resources.avantes.avaspec.Avantes.SensType attribute), 103 (msl.equipment.resources.avantes.avaspec.SensType attribute), 92  
 SENS\_HAMS11071\_2048X64 SENS\_HAMS8378\_256



|                        |  |
|------------------------|--|
|                        | ( <i>msl.equipment.resources.avantes.avaspec.Avantes.SensType</i> ), 578   |
|                        | attribute), 103  |
| SENS_HAMS8378_256      | SensType (class in ( <i>msl.equipment.resources.avantes.avaspec.SensType</i> <i>msl.equipment.resources.avantes.avaspec</i> ), attribute), 92            |
| SENS_HAMS8378_512      | serial ( <i>msl.equipment.connection_serial.ConnectionSerial</i> ( <i>msl.equipment.resources.avantes.avaspec.Avantes.SensType</i> ), 34 attribute), 103 |
| SENS_HAMS8378_512      | SERIAL ( <i>msl.equipment.constants.Interface</i> attribute), 44   |
|                        | ( <i>msl.equipment.resources.avantes.avaspec.SensType</i> ( <i>msl.equipment.record_types.ConnectionRecord</i> attribute), 92                            |
| SENS_HAMS9201          | serial ( <i>msl.equipment.record_types.EquipmentRecord</i> attribute), 103   |
| SENS_HAMS9201          | serial ( <i>msl.equipment.resources.avantes.avaspec.Avantes.Broadcast</i> attribute), 92   |
| SENS_HAMS9840          | serial ( <i>msl.equipment.resources.avantes.avaspec.BroadcastAnswer</i> attribute), 103  |
| SENS_HAMS9840          | serial ( <i>msl.equipment.resources.avantes.avaspec.BroadcastAnswer</i> attribute), 94   |
| SENS_HAMS9840          | serial ( <i>msl.equipment.resources.avantes.avaspec.BroadcastAnswer</i> attribute), 92   |
| SENS_ILX511            | ( <i>msl.equipment.resources.avantes.avaspec.Avantes.SensType</i> attribute), 103  |
| SENS_ILX511            | SERIAL_NUMBER_BUFFER_SIZE ( <i>msl.equipment.resources.energetiq.eq99.EQ99</i> attribute), 92  |
| SENS_ILX511            | ( <i>msl.equipment.resources.avantes.avaspec.SensType</i> ( <i>msl.equipment.resources.thorlabs.kinesis.motion_control</i> attribute), 93                |
| SENS_ILX554            | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.TLI_D</i> attribute), 103   |
| SENS_ILX554            | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.TLI_D</i> attribute), 555   |
| SENS_ILX554            | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.TLI_D</i> attribute), 92  |
| SENS_SU256LSB          | serial ( <i>msl.equipment.resources.avantes.avaspec.AvsIdent</i> attribute), 103   |
| SENS_SU256LSB          | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.M</i> attribute), 93  |
| SENS_SU512LDB          | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.T</i> attribute), 103   |
| SENS_SU512LDB          | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.T</i> attribute), 556   |
| SENS_SU512LDB          | serial ( <i>msl.equipment.resources.thorlabs.kinesis.structs.T</i> attribute), 93  |
| SENS_TCD1304           | session_id ( <i>msl.equipment.hislip.AsyncInitializeResponse</i> attribute), 103   |
| SENS_TCD1304           | session_id ( <i>msl.equipment.hislip.AsyncInitializeResponse</i> attribute), 55  |
| SENS_TCD1304           | session_id ( <i>msl.equipment.hislip.AsyncInitializeResponse</i> attribute), 92  |
| SENS_TSL1301           | set() ( <i>msl.equipment.resources.bentham.benhw32.Bentham32</i> attribute), 103   |
| SENS_TSL1301           | set() ( <i>msl.equipment.resources.bentham.benhw64.Bentham</i> attribute), 103   |
| SENS_TSL1301           | set() ( <i>msl.equipment.resources.bentham.benhw64.Bentham</i> attribute), 92  |
| SENS_TSL1401           | set_acceleration() ( <i>msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX</i> attribute), 103   |
| SENS_TSL1401           | set_acceleration() ( <i>msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX</i> attribute), 92  |
| SENS_TSL1401           | set_acceleration() ( <i>msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX</i> attribute), 92  |
| SENSOR_STATE_CONNECTED | set_acceleration() ( <i>msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX</i> attribute), 279   |
| SensorMode             | (class in ( <i>msl.equipment.resources.mks_instruments.pr4000b.PR4000b</i> method), 148  |

|  |   |
|--|---|
| <code>set_adaptive_integration_time()</code><br>( <code>msl.equipment.resources.optronic_laboratories.ol756.ol756</code><br>method), 211                   | <code>set_brightness()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 136   |
| <code>set_address()</code> ( <code>msl.equipment.resources.mks_instr.benchtop_4000a.P4000B</code><br>method), 148  | <code>set_burn_in_time()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors</code><br>method), 481             |
| <code>set_adv_trigger_channel_conditions()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k</code><br>method), 318 | <code>set_calibration_file()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors</code><br>method), 481         |
| <code>set_adv_trigger_channel_directions()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k</code><br>method), 317 | <code>set_callback_device_status()</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000.PicoScope4000</code><br>method), 338                           |
| <code>set_adv_trigger_channel_properties()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k</code><br>method), 318 | <code>set_callback_port_status()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.BenchtopStepperMotor</code><br>method), 407          |
| <code>set_adv_trigger_delay()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k</code><br>method), 317              | <code>set_callback_register_status()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors</code><br>method), 481 |
| <code>set_alarm()</code> ( <code>msl.equipment.resources.electron_dynamics.tc_series.TCSeries</code><br>method), 131                                       | <code>set_channel()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.KCubeStepperMotor</code><br>method), 504                             |
| <code>set_analog_out()</code><br>( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>method), 117  | <code>set_channel_led()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.KCubeStepperMotor</code><br>method), 538                         |
| <code>set_averaging_number_of_scan()</code><br>( <code>msl.equipment.resources.optronic_laboratories.ol756.ol756</code><br>method), 211                    | <code>set_control()</code> ( <code>msl.equipment.resources.electron_dynamics.tc_series.TCSeries</code><br>method), 131  |
| <code>set_backlash()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors.BenchtopStepperMotor</code><br>method), 407        | <code>set_current()</code> ( <code>msl.equipment.resources.optronic_laboratories.ol756.ol756</code><br>method), 217   |
| <code>set_backlash()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors</code><br>method), 481   | <code>set_current_limit()</code><br>( <code>msl.equipment.resources.aim_tti.mx_series.MXSeries</code><br>method), 89  |
| <code>set_backlash()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.KCubeStepperMotor</code><br>method), 504              | <code>set_current_meter_averaging()</code><br>( <code>msl.equipment.resources.aim_tti.mx_series.MXSeries</code><br>method), 89  |
| <code>set_bandwidth_filter()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope2k3k.PicoScope2k3k</code><br>method), 323               | <code>set_current_step_size()</code><br>( <code>msl.equipment.resources.aim_tti.mx_series.MXSeries</code><br>method), 89  |
| <code>set_beep()</code> ( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 136   | <code>set_cycle_params()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motors.KCubeStepperMotor</code><br>method), 538                     |

(msl.equipment.resources.thorlabs.kinesis.kcube\_solenoide method), 520  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoide method), 382  
 set\_cycle\_params\_block() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoide method), 521  
 set\_dark\_current\_params() (msl.equipment.resources.optronic\_laboratories.ol756 method), 211  
 set\_data\_buffer() (msl.equipment.resources.picotech.picoscope.picoscope method), 323  
 set\_data\_buffer\_bulk() (msl.equipment.resources.picotech.picoscope.picoscope method), 324  
 set\_data\_buffer\_with\_mode() (msl.equipment.resources.picotech.picoscope.ps4000 method), 338  
 set\_data\_buffers() (msl.equipment.resources.picotech.picoscope.picoscope method), 324  
 set\_data\_buffers\_bulk() (msl.equipment.resources.picotech.picoscope.ps6000 method), 344  
 set\_data\_buffers\_with\_mode() (msl.equipment.resources.picotech.picoscope.ps4000 method), 338  
 set\_dcpid\_params() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo method), 504  
 set\_dead\_band() (msl.equipment.resources.mks\_instruments.pr4000b method), 148  
 set\_det\_bipolar() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 381  
 set\_det\_current() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 381  
 set\_det\_hv\_off() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 382  
 set\_det\_hv\_on() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 382  
 set\_det\_hv\_volts() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 382  
 set\_det\_num\_avg\_read() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 382  
 set\_det\_photon() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper method), 508  
 set\_det\_range() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 382  
 set\_det\_type() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 383  
 set\_det\_unipolar() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 383  
 set\_det\_voltage() (msl.equipment.resources.princeton\_instruments.arc\_instruments method), 383  
 set\_device\_resolution() (msl.equipment.resources.picotech.picoscope.ps4000 method), 343  
 set\_dialog() (msl.equipment.resources.mks\_instruments.pr4000 method), 148  
 set\_digital\_analog\_trigger\_operand() (msl.equipment.resources.picotech.picoscope.ps2000a method), 336  
 set\_digital\_out() (msl.equipment.resources.avantes.avaspec.Avantest method), 1400  
 set\_digital\_outputs() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 508  
 set\_digital\_outputs() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo method), 505  
 set\_digital\_outputs() (msl.equipment.resources.thorlabs.kinesis.kcube\_solenoide method), 512  
 set\_digital\_outputs() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper method), 519  
 set\_digital\_port() (msl.equipment.resources.picotech.picoscope.picoscope\_a method), 372  
 set\_direction() (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper method), 508  
 set\_direction() (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper method), 482  
 set\_direction() (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo method), 501  
 set\_direction() (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper method), 508

method), 539  
set\_display\_text()  
    (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b) method), 149  
set\_encoder\_counter()  
    (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor) method), 409  
set\_encoder\_counter()  
    (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo) method), 505  
set\_encoder\_counter()  
    (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor) method), 539  
set\_ets()  
    (msl.equipment.resources.picotech.picoscope.picoscope\_api) method), 318  
set\_ets()  
    (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor) method), 324  
set\_ets\_time\_buffer()  
    (msl.equipment.resources.picotech.picoscope.picoscope\_api) method), 324  
set\_ets\_time\_buffers()  
    (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo) method), 324  
set\_exception\_class()  
    (msl.equipment.connection.Connection) method), 21  
set\_exposure\_mode()  
    (msl.equipment.resources.energetiq.e99.E99) method), 138  
set\_exposure\_time()  
    (msl.equipment.resources.energetiq.e99.E99) method), 138  
set\_ext\_trigger\_range()  
    (msl.equipment.resources.picotech.picoscope.picoscope\_api) method), 338  
set\_external\_clock()  
    (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo) method), 344  
set\_external\_input\_range()  
    (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b) method), 149  
set\_external\_output\_range()  
    (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b) method), 149  
set\_filter\_position()  
    (msl.equipment.resources.princeton\_instruments.on\_instruments) method), 383  
set\_formula\_relay()  
    (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b) method), 149  
set\_formula\_temporary()  
    (msl.equipment.resources.mks\_instruments.pr4000b.PR4000b) method), 150



|   |  |
|---|--|
| set_io_settings()<br>(msl.equipment.resources.thorlabs.kinesis.filter_flipper.FlippedFilterFlipper<br>method), 465                    | set_jog_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor<br>method), 540                  |
| set_ip_details()<br>(msl.equipment.resources.picotech.pt104.PT104<br>method), 360   | set_joystick_params()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 410           |
| set_jog_mode()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 409               | set_lamptime()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 138                  |
| set_jog_mode()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 483         | set_led()<br>(msl.equipment.resources.picotech.picoscope.ps2000.PS2000<br>method), 330   |
| set_jog_mode()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 506                               | set_led_switches()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 484        |
| set_jog_mode()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor<br>method), 540                     | set_led_switches()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 507                              |
| set_jog_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 410       | set_legacy_bus_scanning()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoid_valve.KCubeSolenoidValve<br>method), 507           |
| set_jog_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 483 | set_light()<br>(msl.equipment.resources.picotech.picoscope.ps2000.PS2000<br>method), 330   |
| set_jog_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 506                       | set_limit_switch()<br>(msl.equipment.resources.mks_instruments.pr4000b.PR4000b<br>method), 150   |
| set_jog_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor<br>method), 540             | set_limit_switch()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 411              |
| set_jog_step_size()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 410          | set_limit_switch()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 484        |
| set_jog_step_size()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 483    | set_limit_switch_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 507                       |
| set_jog_step_size()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 506                          | set_limit_switch_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor<br>method), 541             |
| set_jog_step_size()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor<br>method), 540                | set_limit_switch_params()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 411       |
| set_jog_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.BenchtopStepperMotor<br>method), 410         | set_limit_switch_params()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 485 |
| set_jog_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors.IntegratedStepperMotors<br>method), 484   | set_limit_switch_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 507                       |
| set_jog_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDCServo<br>method), 506                         | set_limit_switch_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor<br>method), 540       |

```

method), 541
set_limits_software_approach_policy() (msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo
(msl.equipment.resources.thorlabs.kinesis.benchtop_stepped_motor.BenchtopStepperMotor
method), 411
set_limits_software_approach_policy() (msl.equipment.resources.thorlabs.kinesis.kcube_solenoid
(msl.equipment.resources.thorlabs.kinesis.integrated_stepped_motors.IntegratedStepperMotors
method), 485
set_limits_software_approach_policy() (msl.equipment.resources.thorlabs.kinesis.kcube_stepper_
(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KC254DCServo
method), 507
set_limits_software_approach_policy() method), 193
(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KC254StepperMotor
method), 541
set_linearization_point() (msl.equipment.resources.princeton_instruments.arc_instr
method), 383
(msl.equipment.resources.mks_instruments.set1000nBR4000Br_position())
method), 150
set_linearization_size() (msl.equipment.resources.princeton_instruments.arc_instr
method), 384
(msl.equipment.resources.mks_instruments.set1000nBR4000Brng())
method), 150
set_lower_limit() (msl.equipment.resources.princeton_instruments.arc_instr
method), 384
(msl.equipment.resources.mks_instruments.set1000nBR4000Brng_gadjust())
method), 150
set_mains() (msl.equipment.resources.picotech.pt104.PT104 method), 384
method), 360
set_max_velocity() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.fwx2c.FilterWheelIXX2C), 384
method), 581
set_message_buffer() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.energetiq.eq99.EQ99 method), 384
method), 140
set_min_velocity() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.fwx2c.FilterWheelIXX2C), 384
method), 581
set_mmi_params() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KC254DCServo
method), 508
set_mmi_params() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KC254Solenoid
method), 521
set_mmi_params() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KC254StepperMotor
method), 542
set_mmi_params_block() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KC254DCServo
method), 508
set_mmi_params_block() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.kinesis.kcube_solenoid.KC254Solenoid
method), 521
set_mmi_params_block() (msl.equipment.resources.princeton_instruments.arc_instr
(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KC254StepperMotor
method), 542
set_mmi_params_block() (msl.equipment.resources.princeton_instruments.arc_instr
set_mmi_turret()

```

```

set_mono_wavelength_abs_cm((msl.equipment.resources.princeton_instruments.arc
method), 385 (msl.equipment.princeton_instruments.arc
method), 509
set_mono_wavelength_ang((msl.equipment.resources.princeton_instruments.arc
method), 386 (msl.equipment.princeton_instruments.arc
method), 543
set_mono_wavelength_ev((msl.equipment.resources.princeton_instruments.arc
method), 386 (msl.equipment.princeton_instruments.arc
method), 412
set_mono_wavelength_micron((msl.equipment.resources.princeton_instruments.arc
method), 386 (msl.equipment.princeton_instruments.arc
method), 510
set_mono_wavelength_nm((msl.equipment.resources.princeton_instruments.arc
method), 386 (msl.equipment.princeton_instruments.arc
method), 544
set_mono_wavelength_rel_cm((msl.equipment.resources.princeton_instruments.arc
method), 386 (msl.equipment.princeton_instruments.arc
method), 413
set_motor_params((msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors
method), 412 (msl.equipment.thorlabs.kinesis.benchtop_stepper_motors
method), 486
set_motor_params((msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors
method), 485 (msl.equipment.thorlabs.kinesis.integrated_stepper_motors
method), 510
set_motor_params((msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo
method), 509 (msl.equipment.thorlabs.kinesis.kcube_dc_servo
method), 544
set_motor_params((msl.equipment.resources.thorlabs.kinesis.kcube_stepper
method), 543 (msl.equipment.thorlabs.kinesis.kcube_stepper
method), 413
set_motor_params_ext((msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors
method), 412 (msl.equipment.thorlabs.kinesis.benchtop_stepper_motors
method), 486
set_motor_params_ext((msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors
method), 485 (msl.equipment.thorlabs.kinesis.integrated_stepper_motors
method), 510
set_motor_params_ext((msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo
method), 509 (msl.equipment.thorlabs.kinesis.kcube_dc_servo
method), 544
set_motor_params_ext((msl.equipment.resources.thorlabs.kinesis.kcube_stepper
method), 543 (msl.equipment.thorlabs.kinesis.kcube_stepper
method), 413
set_motor_travel_limits((msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motors
method), 412 (msl.equipment.thorlabs.kinesis.benchtop_stepper_motors
method), 487
set_motor_travel_limits((msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motors
method), 486 (msl.equipment.thorlabs.kinesis.integrated_stepper_motors
method), 510
set_motor_travel_limits((msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo
method), 509 (msl.equipment.thorlabs.kinesis.kcube_dc_servo
method), 544

```

```

        (msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor
        method), 544
        set_output_drive()
set_multi_off_action() (msl.equipment.resources.electron_dynamics.tc_series.TC
        (msl.equipment.resources.aim_tti.mx_series.MXSeries method), 133
        method), 90
        set_output_range()
set_multi_off_delay() (msl.equipment.resources.mks_instruments.pr4000b.PR4000B
        (msl.equipment.resources.aim_tti.mx_series.MXSeries method), 151
        method), 90
        set_output_status()
set_multi_on_action() (msl.equipment.resources.optosigma.shot702.SHOT702
        (msl.equipment.resources.aim_tti.mx_series.MXSeries method), 194
        method), 89
        set_over_current_protection()
set_multi_on_delay() (msl.equipment.resources.aim_tti.mx_series.MXSeries
        (msl.equipment.resources.aim_tti.mx_series.MXSeries method), 90
        method), 90
        set_over_voltage_protection()
set_ncl_filter_position() (msl.equipment.resources.aim_tti.mx_series.MXSeries
        (msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments
        method), 386
        set_parameter()
set_ncl_filter_present() (msl.equipment.resources.avantes.avaspec.Avantes
        (msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments
        method), 386
        set_pid_loop_encoder_coeff()
set_ncl_shutter_closed() (msl.equipment.resources.thorlabs.kinesis.benchtop_stepp
        (msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments
        method), 387
        set_pid_loop_encoder_coeff()
set_ncl_shutter_open() (msl.equipment.resources.thorlabs.kinesis.kcube_stepper_
        (msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments
        method), 387
        set_pid_loop_encoder_params()
set_ncl_ttl_out_off() (msl.equipment.resources.thorlabs.kinesis.benchtop_stepp
        (msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments
        method), 387
        set_pid_loop_encoder_params()
set_ncl_ttl_out_on() (msl.equipment.resources.thorlabs.kinesis.kcube_stepper_
        (msl.equipment.resources.princeton_instruments.arc_nishadyeh1.PrincetonInstruments
        method), 387
        set_pmt_flux_overload_voltage()
set_no_of_captures() (msl.equipment.resources.optronic_laboratories.ol756ocx
        (msl.equipment.resources.picotech.picoscope.picoscopeapi.PicoScopeApi
        method), 325
        set_pmt_hi_voltage()
set_oem_parameter() (msl.equipment.resources.optronic_laboratories.ol756ocx
        (msl.equipment.resources.avantes.avaspec.Avantes method), 213
        method), 119
        set_position()
set_offset() (msl.equipment.resources.mks_instruments.pr4000b.PR4000B
        (msl.equipment.resources.thorlabs.fwx2c.FilterWheelXX
        method), 150
        method), 581
set_operating_mode() set_position_count()
        (msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.KCubeSolenoide
        method), 522
        method), 582
set_operating_state() set_position_counter()
        (msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.KCubeSolenoide
        method), 522
        method), 414
set_origin() (msl.equipment.resources.optosigma.shot702.SHOT702
        method), 194
        (msl.equipment.resources.thorlabs.kinesis.integrated_stepp
        method), 133
        set_position_counter()
set_output() (msl.equipment.resources.electron_dynamics.tc_series.TC
        method), 133
set_output() (msl.equipment.resources.energetiq.eq99.EQ99
        msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo

```



---

|              |            |
|--------------|------------|
| <b>Index</b> | <b>725</b> |
|--------------|------------|

|  |  |
|--|--|
| <code>set_settling_time()</code><br>( <code>msl.equipment.resources.optronic_laboratories.ol756c.ol756c</code><br>method), 213         | <code>set_stage_axis_limits()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.integrated_step</code><br>method), 488                      |
| <code>set_setup()</code> ( <code>msl.equipment.resources.optronic_laboratories.ol756c.ol756c</code><br>method), 217                    | <code>set_stage_axis_limits()</code> ( <code>msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo</code><br>method), 511                          |
| <code>set_shutter_init()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 139                          | <code>set_stage_axis_limits()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.kcube_stepper</code><br>method), 546                        |
| <code>set_shutter_state()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 139                         | <code>set_steps()</code> ( <code>msl.equipment.resources.optosigma.shot702.SHOT702</code><br>method), 195  |
| <code>set_sig_gen_arbitrary()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 325            | <code>set_sync_mode()</code><br>( <code>msl.equipment.resources.avantes.avaspec.Avantes</code><br>method), 120                                       |
| <code>set_sig_gen_arbitrary()</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000(PicoScope2000</code><br>method), 330 | <code>set_tab_delimited_mode()</code><br>( <code>msl.equipment.resources.optronic_laboratories.ol756c.ol756c</code><br>method), 214                  |
| <code>set_sig_gen_built_in()</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps2000(PicoScope2000</code><br>method), 330  | <code>set_termination()</code><br>( <code>msl.equipment.resources.energetiq.eq99.EQ99</code><br>method), 140   |
| <code>set_sig_gen_builtin()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 326              | <code>set_test()</code> ( <code>msl.equipment.resources.electron_dynamics.tc_series</code><br>method), 135   |
| <code>set_sig_gen_builtin_v2()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 326           | <code>set_timebase()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 315                                   |
| <code>set_sig_gen_properties_arbitrary()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 327 | <code>set_timeout()</code> ( <code>msl.equipment.hislip.HiSLIPClient</code><br>method), 68   |
| <code>set_sig_gen_properties_builtin()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 327   | <code>set_timeout()</code> ( <code>msl.equipment.vxi11.RPCCClient</code><br>method), 589   |
| <code>set_siggen()</code> ( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 333                          | <code>set_transit_time()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.filter_flipper.FilterFlipper</code><br>method), 465              |
| <code>set_signal_mode()</code><br>( <code>msl.equipment.resources.mks_instruments.pr4000b.PR4000B</code><br>method), 152               | <code>set_trigger()</code> ( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 318                                       |
| <code>set_simple_trigger()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 327               | <code>set_trigger2()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope</code><br>method), 318                                   |
| <code>set_speed()</code> ( <code>msl.equipment.resources.optosigma.shot702.SHOT702</code><br>method), 194                              | <code>set_trigger_channel_conditions()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope_a</code><br>method), 328               |
| <code>set_speed_mode()</code><br>( <code>msl.equipment.resources.thorlabs.fwx2c.FilterWheelX2c</code><br>method), 582                  | <code>set_trigger_channel_conditions_v2()</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps3000a.PicoScope3000a</code><br>method), 336 |
| <code>set_speed_origin()</code><br>( <code>msl.equipment.resources.optosigma.shot702.SHOT702</code><br>method), 195                    | <code>set_trigger_channel_directions()</code><br>( <code>msl.equipment.resources.picotech.picoscope.picoscope_a</code><br>method), 328               |
| <code>set_stage_axis_limits()</code><br>( <code>msl.equipment.resources.thorlabs.kinesis.benchtop_stepper</code><br>method), 415       | <code>set_trigger_channel_directions()</code><br>( <code>msl.equipment.resources.picotech.picoscope.ps4000a.PicoScope4000a</code><br>method), 340    |

|   |  |
|---|--|
| set_trigger_channel_properties()<br>(msl.equipment.resources.picotech.picoscope.picoscope_ps2000a.FluidScope method), 329         | set_trigger_switches()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motor.FluidScope method), 488 |
| set_trigger_config_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDcServo method), 511                 | set_tweak_control()<br>(msl.equipment.resources.mks_instruments.pr4000b.PR4000B method), 152                         |
| set_trigger_config_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.KCubeSolenoid method), 522               | set_upper_limit()<br>(msl.equipment.resources.mks_instruments.pr4000b.PR4000B method), 152                           |
| set_trigger_config_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper.KCubeStepper method), 546                  | set_user_defined_integration_time()<br>(msl.equipment.KCubeStepper.Mutononic_laboratories.ol756ocx method), 214      |
| set_trigger_config_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDcServo method), 512           | set_valves()<br>(msl.equipment.resources.mks_instruments.pr4000b.PR4000B method), 152                                |
| set_trigger_config_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_solenoide.KCubeSolenoid method), 522         | set_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.FluidScope method), 488         |
| set_trigger_config_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper.KCubeStepper method), 546            | set_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motor.FluidScope method), 488       |
| set_trigger_delay()<br>(msl.equipment.resources.picotech.picoscope.picoscope_ps2000a.FluidScope method), 328                      | set_vel_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDcServo method), 513               |
| set_trigger_digital_port_properties()<br>(msl.equipment.resources.picotech.picoscope.ps2000a.FluidScope method), 332              | set_vel_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.FluidScope method), 488      |
| set_trigger_digital_port_properties()<br>(msl.equipment.resources.picotech.picoscope.ps3000a.FluidScope method), 336              | set_vel_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.integrated_stepper_motor.FluidScope method), 489 |
| set_trigger_mode()<br>(msl.equipment.resources.energetiq.eq99.EQ99 method), 139   | set_vel_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDcServo method), 513         |
| set_trigger_mode()<br>(msl.equipment.resources.thorlabs.fwxx2c.FilterWheelMX2C method), 582                                       | set_vel_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.FluidScope method), 488      |
| set_trigger_params_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDcServo method), 512                 | set_voltage()<br>(msl.equipment.resources.aim_tti.mx_series.MXSeries method), 90                                     |
| set_trigger_params_params()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor method), 546       | set_voltage()<br>(msl.equipment.resources.optronic_laboratories.ol756ocx method), 218                                |
| set_trigger_params_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_dc_servo.KCubeDcServo method), 512           | set_voltage_range()<br>(msl.equipment.resources.aim_tti.mx_series.MXSeries method), 91                               |
| set_trigger_params_params_block()<br>(msl.equipment.resources.thorlabs.kinesis.kcube_stepper_motor.KCubeStepperMotor method), 547 | set_voltage_step_size()<br>(msl.equipment.resources.aim_tti.mx_series.MXSeries method), 91                           |
| set_trigger_switches()<br>(msl.equipment.resources.thorlabs.kinesis.benchtop_stepper_motor.FluidScope method), 415                | set_voltage_tracking_mode()<br>(msl.equipment.resources.aim_tti.mx_series.MXSeries method), 91                       |

`set_wattage()` (`msl.equipment.resources.optronic_laboratory.olc.olc_source.OLCurrentSource`  
 method), 218 `SIGNAL_GENERATOR_ARRAY_OF_AWG_WAVEFORM_VALUES`  
`set_waveform_limiter()` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 method), 344 `SIGNAL_GENERATOR_AWG`  
`settings` (`msl.equipment.resources.thorlabs.kinesis.motion_control.motion_control_resources.picotech.picoscope.enums.PS4000A`  
 property), 553 attribute), 284  
`SETTINGS_RESERVED_LEN` `SIGNAL_GENERATOR_AWG_DELTA_PHASE_INCREMENT`  
 (`msl.equipment.resources.avantes.avaspec.Avantes` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 100 attribute), 284  
`settledError` (`msl.equipment.resources.thorlabs.kinesis.struct.kina_211a.RangeParameters`  
 attribute), 559 (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 284  
`settleSamples` (`msl.equipment.resources.thorlabs.kinesis.struct.kina_211a.RangeParameters`  
 attribute), 570 `SIGNAL_GENERATOR_AWG_INDEX_MODE`  
`settleSamples` (`msl.equipment.resources.thorlabs.kinesis.struct.kina_211a.RangeParameters`  
 attribute), 562 attribute), 284  
`SEVEN` (`msl.equipment.constants.DataBits` at- `SIGNAL_GENERATOR_AWG_START_DELTA_PHASE`  
 tribute), 45 (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 284  
`shared_released` `SIGNAL_GENERATOR_AWG_STOP_DELTA_PHASE`  
 (`msl.equipment.hislip.AsyncLockResponse` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 property), 57 attribute), 284  
`SHORT_PRESS` (`msl.equipment.resources.picotech.picoscope.enums.PS3000B.ButtonState`  
 attribute), 228 `SIGNAL_GENERATOR_AWG_WAVEFORM_SIZE`  
`SHOT702` (class in `msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
`msl.equipment.resources.optosigma.shot702`), attribute), 284  
 190 `SIGNAL_GENERATOR_BUILT_IN`  
`SHOT_SWEEP_TRIGGER_CONTINUOUS_RUN` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 331 `SIGNAL_GENERATOR_BUILT_IN_DWELL_TIME`  
`SHOT_SWEEP_TRIGGER_CONTINUOUS_RUN` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 335 `SIGNAL_GENERATOR_BUILT_IN_INCREMENT`  
`SHOT_SWEEP_TRIGGER_CONTINUOUS_RUN` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 342 `SIGNAL_GENERATOR_BUILT_IN_START_FREQUENCY`  
`shutdown_handler()` (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 method), 127 attribute), 284  
`SIA3` (class in `msl.equipment.resources.cmi.sia3`), (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 126 attribute), 284  
`sig_gen_arbitrary_min_max_values()` `SIGNAL_GENERATOR_BUILT_IN_WAVE_TYPE`  
 (`msl.equipment.resources.picotech.picoscope.picoscope.enums.PS4000A`  
 method), 328 attribute), 284  
`sig_gen_frequency_to_phase()` `SIGNAL_GENERATOR_EXT_IN_THRESHOLD`  
 (`msl.equipment.resources.picotech.picoscope.picoscope.enums.PS4000A`  
 method), 328 attribute), 284  
`sig_gen_software_control()` `SIGNAL_GENERATOR_OFFSET_VOLTAGE`  
 (`msl.equipment.resources.picotech.picoscope.picoscope.enums.PS4000A`  
 method), 328 attribute), 284  
`SIGNAL_GENERATOR` `SIGNAL_GENERATOR_OPERATION`  
 (`msl.equipment.resources.picotech.picoscope.enums.PS4000A`  
 attribute), 284



|   |  |
|---|--|
| attribute), 284   | attribute), 341  |
| SIGNAL_GENERATOR_PK_TO_PK<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 284       | SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 342 |
| SIGNAL_GENERATOR_SETTINGS<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 275       | SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 344 |
| SIGNAL_GENERATOR_SHOTS<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 284          | SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000A.PicoScopeValues<br>attribute), 228                  |
| SIGNAL_GENERATOR_SWEEP_TYPE<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 284     | SINE (msl.equipment.resources.picotech.picoscope.enums.PS2000V.PicoScopeValues<br>attribute), 228                  |
| SIGNAL_GENERATOR_SWEEPS<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 284         | SINE (msl.equipment.resources.picotech.picoscope.enums.PS3000A.PicoScopeValues<br>attribute), 253                  |
| SIGNAL_GENERATOR_TRIGGER_SOURCE<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 284 | SINE (msl.equipment.resources.picotech.picoscope.enums.PS3000V.PicoScopeValues<br>attribute), 264                  |
| SIGNAL_GENERATOR_TRIGGER_TYPE<br>(msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 284   | SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000A.PicoScopeValues<br>attribute), 285                  |
| SIGNAL_MODES (msl.equipment.resources.mks_instruments.pra4000b.PicoScopeValues<br>attribute), 142                               | SINE (msl.equipment.resources.picotech.picoscope.enums.PS5000V.PicoScopeValues<br>attribute), 285                  |
| SINC (msl.equipment.resources.picotech.picoscope.enums.PS2000A.PicoScopeValues<br>attribute), 237                               | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.enums.PS2000A.PicoScopeValues<br>attribute), 332 |
| SINC (msl.equipment.resources.picotech.picoscope.enums.PS2000V.PicoScopeValues<br>attribute), 229                               | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScopeValues<br>attribute), 253       |
| SINC (msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 275                               | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScopeValues<br>attribute), 253       |
| SINC (msl.equipment.resources.picotech.picoscope.enums.PS4000V.PicoScopeValues<br>attribute), 264                               | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps4000a.PicoScopeValues<br>attribute), 339       |
| SINC (msl.equipment.resources.picotech.picoscope.enums.PS5000A.PicoScopeValues<br>attribute), 297                               | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps5000.PicoScopeValues<br>attribute), 341        |
| SINC (msl.equipment.resources.picotech.picoscope.enums.PS5000V.PicoScopeValues<br>attribute), 288                               | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScopeValues<br>attribute), 342       |
| SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps2000a.PicoScopeValues<br>attribute), 332                    | SINE_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps6000.PicoScopeValues<br>attribute), 344        |
| SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScopeValues<br>attribute), 335                    | SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS2000A.PicoScopeValues<br>attribute), 230                |
| SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps3000a.PicoScopeValues<br>attribute), 339                    | SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS3000A.PicoScopeValues<br>attribute), 255                |
| SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScopeValues<br>attribute), 339                    | SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS4000A.PicoScopeValues<br>attribute), 277                |
| SINC_MAX_FREQUENCY<br>(msl.equipment.resources.picotech.picoscope.ps5000a.PicoScopeValues<br>attribute), 339                    | SINGLE (msl.equipment.resources.picotech.picoscope.enums.PS4000V.PicoScopeValues<br>attribute), 250                |

---

730
Index

SQUARE (*msl.equipment.resources.picotech.picoscope.enums.PS5000WaveType* attribute), 297

SQUARE (*msl.equipment.resources.picotech.picoscope.structs.PS5000WaveType* attribute), 288

SQUARE (*msl.equipment.resources.picotech.picoscope.enums.PS6000WaveType* attribute), 306

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 513

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.picotech.picoscope.ps2000a.PhaseScope* attribute), 332

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 69

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.picotech.picoscope.ps3000a.PhaseScope* attribute), 335

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 62

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.picotech.picoscope.ps4000a.PhaseScope* attribute), 339

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 62

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.picotech.picoscope.ps5000a.PhaseScope* attribute), 341

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 62

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.picotech.picoscope.ps5000a.PhaseScope* attribute), 342

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 218

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.picotech.picoscope.ps6000a.PhaseScope* attribute), 344

SQUARE\_MAX\_FREQUENCY (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_method*), 62

SS\_TRIGGER\_MODE (*msl.equipment.resources.avantes.avaspec.AvsStatus* attribute), 101

SS\_TRIGGER\_MODE (*msl.equipment.resources.avantes.avaspec.AvsStatus* attribute), 104

stageID (*msl.equipment.resources.thorlabs.kinesis.status.MOT.StepAxisParameters* attribute), 558

stageID (*msl.equipment.resources.thorlabs.kinesis.status.MOT.StepAxisParameters* attribute), 93

StandAloneType (*class in msl.equipment.resources.avantes.avaspec*), 97

StandAloneType (*msl.equipment.resources.avantes.avaspec.BroadcastAnswer* attribute), 94

start() (*msl.equipment.resources.dataray.datarayocx\_32.DataRayOCX32* method), 127

start() (*msl.equipment.resources.dataray.datarayocx\_64.DataRayOCX64* method), 130

start\_log() (*msl.equipment.resources.bentham.benhw32.Benthaw32* method), 124

start\_log() (*msl.equipment.resources.bentham.benhw32.Benthaw32* method), 152

start\_logging() (*msl.equipment.resources.omega.ithx.iTHX* method), 189

start\_logging() (*msl.equipment.resources.omega.ithx.iTHX* method), 196

start\_polling() (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepped\_motor.BenchtopStepperMotor* method), 416

start\_polling() (*msl.equipment.resources.dataray.datarayocx\_32.DataRayOCX32* method), 130

start\_polling() (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepped\_motor.BenchtopStepperMotor* method), 130

start\_polling() (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepped\_motor.BenchtopStepperMotor* method), 466

start\_polling() (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepped\_motor.BenchtopStepperMotor* method), 196

start\_polling() (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepped\_motors.IntegratedStepperMotors* method), 489

start\_polling() (*msl.equipment.resources.thorlabs.kinesis.integrated\_stepped\_motors.IntegratedStepperMotors* method), 489

start\_polling() (*msl.equipment.connection\_serial.ConnectionSerial* property), 34

`stop_immediate()` (method), 196  
 (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor), 44  
 (method), 417  
`stop_immediate()` (attribute), 556  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motor), 44  
 (method), 489  
 (attribute), 435  
`stop_immediate()` (stream\_readers)  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo), 26  
 (method), 513  
 (property), 26  
`stop_immediate()` (stream\_writers)  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor), 44  
 (method), 548  
 (property), 26  
`stop_log()` (msl.equipment.resources.bentham.benchtop\_stepper\_motor), 44  
 (method), 124  
 (msl.equipment.resources.picotech.picoscope.ps3000.PicoScope), 44  
`stop_measure()` (method), 334  
 (msl.equipment.resources.avantes.avaspec.attenuator), 83  
 (method), 120  
`stop_measurement()` (success (msl.equipment.hislip.AsyncEndTLSResponse)  
 (msl.equipment.resources.optronic\_laboratories.ol750scope), 1556  
 (method), 214  
 (success (msl.equipment.hislip.AsyncLockResponse  
 (property), 57  
`stop_polling()` (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor), 44  
 (method), 417  
 (property), 63  
`stop_polling()` (success (msl.equipment.hislip.AuthenticationResult  
 (msl.equipment.resources.thorlabs.kinesis.filter\_flipper\_flipper), 67  
 (method), 466  
 (SUCCESS (msl.equipment.vxi11.AcceptStatus attribute), 587  
`stop_polling()` (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motor), 44  
 (method), 489  
 (attribute), 575  
`stop_polling()` (suppress\_stray\_light()  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo), 26  
 (method), 513  
 (msl.equipment.resources.avantes.avaspec.Avantest), 120  
`stop_polling()` (suspend\_move\_messages()  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_solenooid), 44  
 (method), 523  
 (method), 417  
`stop_polling()` (suspend\_move\_messages()  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor), 44  
 (method), 548  
 (method), 514  
`stop_profiled()` (suspend\_move\_messages()  
 (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor), 44  
 (method), 417  
 (method), 548  
`stop_profiled()` (SW\_TRIGGER\_MODE  
 (msl.equipment.resources.thorlabs.kinesis.integrated\_stepper\_motor), 44  
 (method), 489  
 (attribute), 101  
`stop_profiled()` (SwitchBreaks (msl.equipment.resources.thorlabs.kinesis.enums.Kinesis), 44  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo), 26  
 (method), 513  
 (SwitchBreaks\_HomeOnly  
 (msl.equipment.resources.thorlabs.kinesis.enums.KIM\_Library), 44  
 (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor), 44  
 (method), 548  
 (SwitchMakes (msl.equipment.resources.thorlabs.kinesis.enums.Kinesis), 44  
`stop_slowly()` (msl.equipment.resources.optosigma.shot702s), 440



**SwitchMakes\_HomeOnly** (class in `msl.equipment.resources.thorlabs.kinesis.enums.KinesisSwitchModes`), 440  
**TC\_LoopParameters** (class in `msl.equipment.resources.thorlabs.kinesis.structs`), 578  
**SWOnly** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_SensorTypes` attribute), 434  
**TC\_SensorTypes** (class in `msl.equipment.resources.thorlabs.kinesis.enums`), 461  
**SYNC\_TRIGGER** (`msl.equipment.resources.avantes.avaspec.Avantes` attribute), 101  
**TC\_TargetTemperature** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**SyncClient** (class in `msl.equipment.hislip`), 68  
**synchronous** (`msl.equipment.connection_tcpip_hislip.ConnectionTCP_HISLIP` property), 37  
**TC\_TempDifference** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**system** (`msl.equipment.connection_nidaq.ConnectionNIDAQ` property), 26  
**TC\_TH200kOhm** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**system\_status\_byte** (`msl.equipment.resources.optronic_laboratories.ol_current_source_OLCurrentSource` property), 218  
**TC\_TH20kOhm** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**T**  
**TC\_Transducer** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**TAGS** (`msl.equipment.resources.mks_instruments.pr4000b.PR4000B` attribute), 142  
**TCD\_FIRST\_USED\_DARK\_PIXEL** (`msl.equipment.resources.avantes.avaspec.Avantes` attribute), 175  
**take\_point\_to\_point\_calibration()** (`msl.equipment.resources.optronic_laboratories.ol756module.OL756` method), 214  
**TCD\_TOTAL\_DARK\_PIXELS** (`msl.equipment.resources.avantes.avaspec.Avantes` attribute), 175  
**take\_point\_to\_point\_measurement()** (`msl.equipment.resources.optronic_laboratories.ol756module.OL756` method), 214  
**TCD\_USED\_DARK\_PIXELS** (`msl.equipment.resources.avantes.avaspec.Avantes` attribute), 175  
**take\_quick\_scan\_calibration()** (`msl.equipment.resources.optronic_laboratories.ol756module.OL756` method), 215  
**TCPIP\_HISLIP** (`msl.equipment.constants.Interface` attribute), 44  
**take\_quick\_scan\_measurement()** (`msl.equipment.resources.optronic_laboratories.ol756module.OL756` method), 215  
**TC\_SERIES** (`msl.equipment.constants.Interface` attribute), 44  
**target\_info()** (`msl.equipment.resources.optronic_laboratories.ol_current_source_OLCurrentSource` method), 218  
**msl.equipment.resources.electron\_dynamics.tc\_series**, 130  
**Task** (`msl.equipment.connection_nidaq.ConnectionNIDAQ` property), 26  
**TCube\_Brushless\_Motor** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 552  
**TBD1** (`msl.equipment.resources.thorlabs.kinesis.enums.TST_Stages` attribute), 461  
**TCube\_Servo** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 552  
**TBD2** (`msl.equipment.resources.thorlabs.kinesis.enums.TST_Stages` attribute), 461  
**TCube\_Inertial\_Motor** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 552  
**TBD3** (`msl.equipment.resources.thorlabs.kinesis.enums.TST_Stages` attribute), 461  
**TCube\_LaserDiode** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 553  
**TBD4** (`msl.equipment.resources.thorlabs.kinesis.enums.TST_Stages` attribute), 461  
**TC\_ActualTemperature** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**TCube\_LaserDiode** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 553  
**TC\_Current** (`msl.equipment.resources.thorlabs.kinesis.enums.TC_DisplayModes` attribute), 462  
**TCube\_LaserDiode** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 552  
**TC\_DisplayModes** (class in `msl.equipment.resources.thorlabs.kinesis.enums`), 462  
**TCube\_NanoTrak** (`msl.equipment.resources.thorlabs.kinesis.motion_control` attribute), 552

attribute), 553  
 TCube\_Quad (msl.equipment.resources.thorlabs.kinesis.motion\_control.modules.motion\_control.avantes.avaspec),  
 attribute), 553  
 TCube\_Solenoid (msl.equipment.resources.thorlabs.kinesis.motion\_control.modules.motion\_control), 586  
 attribute), 553  
 TCube\_Stepper\_Motor (msl.equipment.resources.thorlabs.kinesis.motion\_control.modules.motion\_control), 232  
 attribute), 553  
 TCube\_Strain\_Gauge (msl.equipment.resources.thorlabs.kinesis.motion\_control.modules.motion\_control), 566  
 attribute), 553  
 TCube\_TEC (msl.equipment.resources.thorlabs.kinesis.motion\_control.modules.motion\_control.picotech.picoscope.structs.PS2000A), 347  
 attribute), 553  
 team (msl.equipment.record\_types.EquipmentRecord), 74  
 attribute), 74  
 TEC\_ID (msl.equipment.resources.avantes.avaspec.Avantes), 102  
 attribute), 102  
 TecControlType (class in msl.equipment.resources.avantes.avaspec), 97  
 TECctlr (in module msl.equipment.resources.thorlabs.kinesis.messages), attribute), 352  
 550  
 temperature() (msl.equipment.resources.omega.ithx.iTHX), 187  
 method), 187  
 temperature() (msl.equipment.resources.raicol.raicol), 388  
 method), 388  
 temperature\_humidity() (msl.equipment.resources.omega.ithx.iTHX), 188  
 method), 188  
 temperature\_humidity\_dewpoint() (msl.equipment.resources.omega.ithx.iTHX), 188  
 method), 188  
 TEMPERATURE\_SENSOR (msl.equipment.resources.picotech.picoscope.structs.PS4000PHYS), 262  
 attribute), 262  
 TEMPERATURE\_UPTO\_100 (msl.equipment.resources.picotech.picoscope.structs.PS4000PHYS), 261  
 attribute), 261  
 TEMPERATURE\_UPTO\_130 (msl.equipment.resources.picotech.picoscope.structs.PS4000PHYS), 261  
 attribute), 261  
 TEMPERATURE\_UPTO\_40 (msl.equipment.resources.picotech.picoscope.structs.PS4000PHYS), 261  
 attribute), 261  
 TEMPERATURE\_UPTO\_70 (msl.equipment.resources.picotech.picoscope.structs.PS4000PHYS), 261  
 attribute), 261  
 TEMPERATURES (msl.equipment.resources.picotech.picoscope.structs.PS4000ChannelInfo), 262  
 attribute), 262

(msl.equipment.resources.picotech.picoscope.structs.PS1000TriggerChannelProperties attribute), 354  
 (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 350  
 thresholdMinor thresholdUpperHysteresis  
 (msl.equipment.resources.picotech.picoscope.structs.PS1000TriggerChannelProperties attribute), 345  
 (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 353  
 thresholdMinor thresholdUpperHysteresis  
 (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 347  
 (msl.equipment.resources.picotech.picoscope.structs.PS4000TriggerChannelProperties attribute), 352  
 thresholdMinor thresholdUpperHysteresis  
 (msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerChannelProperties attribute), 354  
 (msl.equipment.resources.picotech.picoscope.structs.PS6000TriggerChannelProperties attribute), 356  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS2000ATriggerChannelProperties attribute), 347  
 msl.equipment.resources.thorlabs.kinesis.structs),  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 345  
 TIM\_ButtonsMode (class in  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerChannelProperties attribute), 350  
 459  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 347  
 msl.equipment.resources.thorlabs.kinesis.enums),  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProperties attribute), 353  
 TIM\_Direction (class in  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS4000ATriggerChannelProperties attribute), 352  
 460  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS5000ATriggerChannelProperties attribute), 356  
 msl.equipment.resources.thorlabs.kinesis.structs),  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS5000TriggerChannelProperties attribute), 354  
 TIM\_JogMode (class in  
 thresholdMode (msl.equipment.resources.picotech.picoscope.structs.PS6000ATriggerChannelProperties attribute), 357  
 459  
 thresholdUpper TIM\_JogParameters (class in  
 (msl.equipment.resources.picotech.picoscope.structs.PS2000TriggerChannelProperties attribute), 347  
 577  
 thresholdUpper TIM\_Status (class in  
 (msl.equipment.resources.picotech.picoscope.structs.PS3000TriggerChannelProperties attribute), 350  
 577  
 thresholdUpper TIME (msl.equipment.resources.picotech.picoscope.enums.PS2000ATriggerChannelProperties attribute), 353  
 TIME (msl.equipment.resources.picotech.picoscope.enums.PS3000ATriggerChannelProperties attribute), 259  
 thresholdUpper TIME (msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerChannelProperties attribute), 352  
 TIME (msl.equipment.resources.picotech.picoscope.enums.PS4000ATriggerChannelProperties attribute), 268  
 thresholdUpper time (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_BrushlessMotor attribute), 356  
 TIME\_1 (msl.equipment.resources.cmi.sia3.IntegrationTime attribute), 126  
 thresholdUpper TIME\_100 (msl.equipment.resources.cmi.sia3.IntegrationTime attribute), 357  
 TIME\_100 (msl.equipment.resources.cmi.sia3.IntegrationTime attribute), 126  
 thresholdUpperHysteresis TIME\_10m (msl.equipment.resources.cmi.sia3.IntegrationTime attribute), 347  
 TIME\_10m (msl.equipment.resources.cmi.sia3.IntegrationTime attribute), 126  
 thresholdUpperHysteresis

**TIME\_1m** (*msl.equipment.resources.cmi.sia3.IntegrationTime* attribute), 126  
**TIME\_2** (*msl.equipment.resources.cmi.sia3.IntegrationTime* attribute), 126  
**TIME\_200m** (*msl.equipment.resources.cmi.sia3.IntegrationTime* attribute), 126  
**TIME\_20m** (*msl.equipment.resources.cmi.sia3.IntegrationTime* attribute), 126  
**TIME\_500m** (*msl.equipment.resources.cmi.sia3.IntegrationTime* attribute), 126  
**TIME\_50u** (*msl.equipment.resources.cmi.sia3.IntegrationTime* attribute), 125  
**time\_since\_last\_msg\_received()** (*msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper\_motor.BenchtopStepperMotor* method), 417  
**time\_since\_last\_msg\_received()** (*msl.equipment.resources.thorlabs.kinesis.filter\_jspike.FilterJspike* method), 466  
**time\_since\_last\_msg\_received()** (*msl.equipment.resources.thorlabs.kinesis.integrated\_steppers.IntegratedStepperMotors* method), 489  
**time\_since\_last\_msg\_received()** (*msl.equipment.resources.thorlabs.kinesis.kcube\_solenoide.KCubeSolenoid* method), 523  
**time\_since\_last\_msg\_received()** (*msl.equipment.resources.thorlabs.kinesis.kcube\_stepper\_motor.KCubeStepperMotor* method), 548  
**timeout** (*msl.equipment.connection\_message\_based\_conversion.MessageBasedConversion* static method), 554  
**timeout** (*msl.equipment.connection\_prologix.ConnectionPrologix* method), 77  
**timeout** (*msl.equipment.resources.thorlabs.kinesis.to\_xml.MotionParameter* method), 80  
**timer()** (*msl.equipment.resources.energetiq.e99.E99xml* method), 141  
**timestampCounter** (*msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerInfo* attribute), 351  
**TimeStampType** (class in *msl.equipment.resources.avantes.avaspec*), 97  
**timeUnits** (*msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerInfo* attribute), 351  
**timeUnits** (*msl.equipment.resources.picotech.picoscope.structs.PS3000ATriggerInfo* attribute), 355  
**TLI\_DeviceInfo** (class in *msl.equipment.resources.thorlabs.kinesis.structs*), 572



`transitTime` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 566  
`trig2Mode` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType` attribute), 237  
`trig2Polarity` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS2000AWaveType` attribute), 228  
`trig2SumMax` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveType` attribute), 253  
`trig2SumMin` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS3000AWaveTypes` attribute), 243  
`Trig_Disabled` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerInPS4000A` attribute), 441  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType` attribute), 274  
`Trig_High` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerInPS4000A` attribute), 441  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType` attribute), 264  
`Trig_In_GPI` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerInPS5000A` attribute), 441  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType` attribute), 297  
`Trig_InAbsoluteMove` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerInPS5000AWaveType` attribute), 441  
`TRIANGLE` (`msl.equipment.resources.picotech.picoscope.enums.PS6000AWaveType` attribute), 306  
`Trig_InRelativeMove` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerInPS6000AWaveType` attribute), 441  
`TRIANGLE_MAX_FREQUENCY` (`msl.equipment.resources.picotech.picoscope.ps2000A` attribute), 332  
`Trig_Out_PosStepFwd` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS2000A` attribute), 441  
`TRIANGLE_MAX_FREQUENCY` (`msl.equipment.resources.picotech.picoscope.ps4000A` attribute), 339  
`Trig_Out_PosStepRev` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS4000A` attribute), 441  
`TRIANGLE_MAX_FREQUENCY` (`msl.equipment.resources.picotech.picoscope.ps5000A` attribute), 341  
`Trig_Out_AtEitherLimit` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS5000A` attribute), 441  
`TRIANGLE_MAX_FREQUENCY` (`msl.equipment.resources.picotech.picoscope.ps5000A` attribute), 342  
`Trig_Out_AtFwdLimit` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS5000A` attribute), 441  
`TRIANGLE_MAX_FREQUENCY` (`msl.equipment.resources.picotech.picoscope.ps6000A` attribute), 344  
`Trig_Out_AtRevLimit` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS6000A` attribute), 441  
`trig1DiffThreshold` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`Trig_Out_GPI` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS4000A` attribute), 441  
`trig1Mode` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`Trig_Out_InMotion` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS4000A` attribute), 441  
`trig1Polarity` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`Trig_Out_PosStepFwd` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS4000A` attribute), 441  
`trig1SumMax` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`Trig_Out_PosStepRev` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS4000A` attribute), 441  
`trig1SumMin` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575  
`Trig_Out_PosStepFwd` (`msl.equipment.resources.thorlabs.kinesis.enums.KIM_TriggerOutPS4000A` attribute), 441  
`trig2DiffThreshold` (`msl.equipment.resources.thorlabs.kinesis.structs.QD_KPA_AutTrigIOConfig` attribute), 575



|  |  |
|--|--|
| <a href="#">attribute</a> ), 282   | <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_</a>                          |
| <a href="#">TRIGGER_PROPERTIES</a>   | <a href="#">attribute</a> ), 568   |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">(msl.equipment.resources.picotech.picoscope.structs</a>                              |
| <a href="#">attribute</a> ), 282   | <a href="#">attribute</a> ), 351   |
| <a href="#">TRIGGER_PROPERTIES_CHANNEL</a>   | <a href="#">triggerTime</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.structs</a> |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">PS4000APicoStringValue</a>   |
| <a href="#">attribute</a> ), 282   | <a href="#">TriggerType</a> (class in  |
| <a href="#">TRIGGER_PROPERTIES_THRESHOLD_LOWER</a>   | <a href="#">msl.equipment.resources.avantes.avaspec</a> ,  |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">PS4000APicoStringValue</a>   |
| <a href="#">attribute</a> ), 282   | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS2000A</a>  |
| <a href="#">TRIGGER_PROPERTIES_THRESHOLD_LOWER_HYSTERESIS</a>  | <a href="#">attribute</a> ), 241   |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS2000A</a>  |
| <a href="#">attribute</a> ), 282   | <a href="#">attribute</a> ), 230   |
| <a href="#">TRIGGER_PROPERTIES_THRESHOLD_MODE</a>  | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS3000A</a>  |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">PS4000APicoStringValue</a>   |
| <a href="#">attribute</a> ), 282   | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS3000A</a>  |
| <a href="#">TRIGGER_PROPERTIES_THRESHOLD_UPPER</a>   | <a href="#">attribute</a> ), 247   |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS4000A</a>  |
| <a href="#">attribute</a> ), 282   | <a href="#">attribute</a> ), 279   |
| <a href="#">TRIGGER_PROPERTIES_THRESHOLD_UPPER_HYSTERESIS</a>  | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS4000A</a>  |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS4000APicoStringValue</a>                               | <a href="#">PS4000APicoStringValue</a>   |
| <a href="#">attribute</a> ), 282   | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS5000A</a>  |
| <a href="#">TRIGGER_RAW</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS6000ASigGenTrigSource</a> | <a href="#">attribute</a> ), 307   |
| <a href="#">attribute</a> ), 307   | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS5000A</a>  |
| <a href="#">trigger_within_pre_trigger_samples()</a>   | <a href="#">attribute</a> ), 291   |
| <a href="#">(msl.equipment.resources.picotech.picoscope.enums.PS6000APicoStringValue</a>                               | <a href="#">TRUE</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS6000A</a>  |
| <a href="#">method</a> ), 328  | <a href="#">attribute</a> ), 309   |
| <a href="#">triggerIndex</a> ( <a href="#">msl.equipment.resources.picotech.picoscope.enums.PS6000ATriggerIndex</a>    | <a href="#">TSG_DisplayModes</a> (class in   |
| <a href="#">attribute</a> ), 355   | <a href="#">msl.equipment.resources.thorlabs.kinesis.enums</a> ),                                |
| <a href="#">TriggerIntervalFwd</a>   | 457  |
| <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams</a>                                   | <a href="#">TSG_FiberTrigParams</a>  |
| <a href="#">attribute</a> ), 568   | <a href="#">attribute</a> ), 458   |
| <a href="#">TriggerIntervalRev</a>   | <a href="#">TSG_Hub_Analogue_Modes</a> (class in   |
| <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams</a>                                   | <a href="#">msl.equipment.resources.thorlabs.kinesis.enums</a> ),                                |
| <a href="#">attribute</a> ), 568   | 457  |
| <a href="#">TriggerMode</a> (class in <a href="#">TSG_HubChannel1</a>  |  |
| <a href="#">msl.equipment.resources.thorlabs.fwx2c</a> ,   | <a href="#">(msl.equipment.resources.thorlabs.kinesis.enums.TSG_Hu</a>                           |
| 579  | <a href="#">attribute</a> ), 457   |
| <a href="#">TriggerPulseCountFwd</a>   | <a href="#">TSG_HubChannel2</a>  |
| <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams</a>                                   | <a href="#">msl.equipment.resources.thorlabs.kinesis.enums.TSG_Hu</a>                            |
| <a href="#">attribute</a> ), 568   | <a href="#">attribute</a> ), 457   |
| <a href="#">TriggerPulseCountRev</a>   | <a href="#">TSG_IOSettings</a> (class in   |
| <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams</a>                                   | <a href="#">msl.equipment.resources.thorlabs.kinesis.structs</a> ),                              |
| <a href="#">attribute</a> ), 568   | 576  |
| <a href="#">TriggerPulseWidth</a>  | <a href="#">TSG_Position</a> ( <a href="#">msl.equipment.resources.thorlabs.kinesis.enums.T</a>  |
| <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams</a>                                   | <a href="#">msl.equipment.resources.thorlabs.kinesis.enums.T</a>                                 |
| <a href="#">attribute</a> ), 568   | <a href="#">TSG_Undefined</a> ( <a href="#">msl.equipment.resources.thorlabs.kinesis.enums</a>   |
| <a href="#">TriggerStartPositionFwd</a>  | <a href="#">attribute</a> ), 458   |
| <a href="#">(msl.equipment.resources.thorlabs.kinesis.structs.KMOT_TriggerParams</a>                                   | <a href="#">TSG_VMOT_TriggerParams</a>   |
| <a href="#">attribute</a> ), 568   | <a href="#">attribute</a> ), 458   |
| <a href="#">TriggerStartPositionRev</a>  | <a href="#">TST_Stages</a> (class in   |

`msl.equipment.resources.thorlabs.kinesis.enums`), `attribute`), 63  
 460 `type (msl.equipment.hislip.AsyncStatusQuery at-`  
`turn_off()` (`msl.equipment.resources.aim_tti.mx_series.MXSeries`), 62  
`method`), 88 `type (msl.equipment.hislip.AsyncStatusResponse`  
`turn_off()` (`msl.equipment.resources.optronic_laboratories.oltronic`), 63  
`method`), 219 `type (msl.equipment.hislip.AuthenticationExchange`  
`turn_off_multi()` `attribute`), 66  
`(msl.equipment.resources.aim_tti.mx_series.MXSeries`), 66  
`method`), 88 `type (msl.equipment.hislip.AuthenticationResult`  
`turn_on()` (`msl.equipment.resources.aim_tti.mx_series.MXSeries`), 66  
`method`), 87 `type (msl.equipment.hislip.AuthenticationStart at-`  
`turn_on()` (`msl.equipment.resources.optronic_laboratories.oltronic`), 63  
`method`), 219 `type (msl.equipment.hislip.DataEnd attribute)`, 56  
`turn_on_multi()` `type (msl.equipment.hislip.DeviceClearAcknowledge`  
`(msl.equipment.resources.aim_tti.mx_series.MXSeries`), 59  
`method`), 87 `type (msl.equipment.hislip.DeviceClearComplete`  
`TWELVE` (`msl.equipment.resources.thorlabs.fwx2c.FilterCount`), 59  
`attribute`), 578 `type (msl.equipment.hislip.EndTLS attribute)`, 64  
`TWO` (`msl.equipment.constants.StopBits attribute`), `type (msl.equipment.hislip.ErrorMessage at-`  
 44 `tribute)`, 54  
`type (msl.equipment.hislip.AsyncDeviceClear at-` `type (msl.equipment.hislip.FatalErrorMessage at-`  
`tribute)`, 58 `tribute)`, 54  
`type (msl.equipment.hislip.AsyncDeviceClearAcknowledge` `type (msl.equipment.hislip.GetDescriptors at-`  
`attribute)`, 59 `tribute)`, 61  
`type (msl.equipment.hislip.AsyncEndTLS at-` `type (msl.equipment.hislip.GetDescriptorsResponse`  
`tribute)`, 64 `attribute)`, 61  
`type (msl.equipment.hislip.AsyncEndTLSResponse` `type (msl.equipment.hislip.GetSaslMechanismList`  
`attribute)`, 64 `attribute)`, 65  
`type (msl.equipment.hislip.AsyncInitialize at-` `type (msl.equipment.hislip.GetSaslMechanismListResponse`  
`tribute)`, 61 `attribute)`, 65  
`type (msl.equipment.hislip.AsyncInitializeResponse` `type (msl.equipment.hislip.Initialize attribute)`, 54  
`attribute)`, 62 `type (msl.equipment.hislip.InitializeResponse at-`  
`type (msl.equipment.hislip.AsyncLock attribute)`, `tribute)`, 55  
 56 `type (msl.equipment.hislip.Message attribute)`, 53  
`type (msl.equipment.hislip.AsyncLockInfo at-` `type (msl.equipment.hislip.StartTLS attribute)`, 63  
`tribute)`, 57 `type (msl.equipment.hislip.Trigger attribute)`, 60  
`type (msl.equipment.hislip.AsyncLockInfoResponse` `type (msl.equipment.record_types.MeasurandRecord`  
`attribute)`, 57 `attribute)`, 77  
`type (msl.equipment.hislip.AsyncLockResponse` `type (msl.equipment.resources.thorlabs.kinesis.structs.TLI_Hardw`  
`attribute)`, 56 `attribute)`, 556  
`type (msl.equipment.hislip.AsyncMaximumMessageSize` `type (msl.equipment.resources.thorlabs.kinesis.structs.TLI_De`  
`attribute)`, 60 `tribute)`, 555  
`type (msl.equipment.hislip.AsyncMaximumMessageSizeResponse` `type (msl.equipment.connection_nidaq.ConnectionNIDAQ`  
`attribute)`, 60 `property)`, 26  
`type (msl.equipment.hislip.AsyncRemoteLocalControl`  
`attribute)`, 58  
`type (msl.equipment.hislip.AsyncRemoteLocalResponse` `type (msl.equipment.resources.nkt.nktpdll.ParameterSetType`  
`attribute)`, 58 `attribute)`, 154  
`type (msl.equipment.hislip.AsyncStartTLS at-` `underOrOverRead`  
`tribute)`, 63 `(msl.equipment.resources.thorlabs.kinesis.structs.KNA_T`  
`type (msl.equipment.hislip.AsyncStartTLSResponse` `tribute)`, 571



[underOrOverRead](#) ([msl.equipment.resources.thorlabs.kinesis.structs.NTAttRead](#) attribute), 563  
[UNIDENTIFIED](#) ([msl.equipment.hislip.ErrorType](#) attribute), 52  
[unique\\_key](#) ([msl.equipment.record\\_types.EquipmentRecord](#) attribute), 74  
[unit](#) ([msl.equipment.record\\_types.MeasurandRecord](#) attribute), 77  
[Unit](#) ([msl.equipment.resources.nkt.nktpdll.ParameterSetType](#) attribute), 154  
[UNIT\\_INFO](#) ([msl.equipment.resources.picotech.picoscope.enums.PS4000AMetaType](#) attribute), 275  
[UnitA](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 160  
[UnitA](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 155  
[UnitCB](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitCB](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[UnitCBm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitCBm](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[UnitcC](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitcC](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[Unitclm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[Unitclm](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[UnitcmA](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitcmA](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[UnitcmW](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 160  
[UnitcmW](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 155  
[UnitcV](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitcV](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[UnitdB](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitdB](#) ([msl.equipment.resources.nkt.nktpdll.ParamSetUnitTypes](#) attribute), 156  
[UnitdBm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitdBm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitdC](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitdC](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[Unitdlm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[Unitdlm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitdmA](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitdmA](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitdmW](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitdmW](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[Unitdnm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[Unitdnm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[Unitdpm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[Unitdpm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitdV](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitdV](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[Unitlm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[Unitlm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitmA](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 160  
[UnitmA](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 155  
[UnitmB](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitmB](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitmBm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitmBm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitmC](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 160  
[UnitmC](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[UnitmCm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[UnitmCm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156  
[Unitmlm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 161  
[Unitmlm](#) ([msl.equipment.resources.nkt.nktpdll.NKT.ParamSetUnitTypes](#) attribute), 156





- property), 26
- ## V
- valid\_det\_num()** (msl.equipment.resources.princeton\_instruments.PrincetonInstruments method), 366
- valid\_enum()** (msl.equipment.resources.princeton\_instruments.PrincetonInstruments static method), 366
- valid\_filter\_enum()** (msl.equipment.resources.princeton\_instruments.PrincetonInstruments static method), 367
- valid\_mono\_enum()** (msl.equipment.resources.princeton\_instruments.PrincetonInstruments static method), 366
- valid\_readout\_enum()** (msl.equipment.resources.princeton\_instruments.arc\_instruments.PrincetonInstruments static method), 367
- value()** (msl.equipment.config.Config method), 19
- VARIANT\_INFO** (msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfoApi attribute), 224
- VARIANT\_INFO** (msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfoApi attribute), 226
- VARIANT\_INFO** (msl.equipment.resources.picotech.picoscope.enums.PicoScopeInfoApi attribute), 244
- VELOCITY** (msl.equipment.resources.thorlabs.kinesis.enums.UniType attribute), 462
- velocity** (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StepParameters attribute), 557
- velocity** (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StepParameters attribute), 567
- velocityFeedForward** (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StepParameters attribute), 559
- velParams** (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StepParameters attribute), 556
- ver()** (msl.equipment.resources.princeton\_instruments.arc\_instruments.PrincetonInstruments static method), 367
- version** (msl.equipment.connection\_nidaq.ConnectionNIDAQ property), 26
- version()** (msl.equipment.connection\_prologix.ConnectionPrologix method), 30
- version()** (msl.equipment.resources.bentham.benhw64.Bentham method), 125
- version()** (msl.equipment.resources.energetiq.e99w4000 method), 141
- version\_info** (in module msl.equipment), 17
- VERSION\_LEN** (msl.equipment.resources.avantes.avaunit1 attribute), 100
- Vertical\_Stage** (msl.equipment.resources.thorlabs.kinesis.structs.MOT\_StepParameters attribute), 553
- verticalComponent** (msl.equipment.resources.thorlabs.kinesis.structs.NT\_HV attribute), 561
- voltage\_offset** (msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel property), 220
- voltage\_range** (msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel property), 222
- VoltageAdjustRate** (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MOT\_StepParameters attribute), 572
- VoltageStep** (msl.equipment.resources.thorlabs.kinesis.structs.KPZ\_MOT\_StepParameters attribute), 572
- volts** (msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel property), 223
- volts\_per\_adu** (msl.equipment.resources.picotech.picoscope.channel.PicoScopeChannel property), 223
- VXIClient** (class in msl.equipment.vxi11), 590
- ## W
- wait\_for\_message()** (msl.equipment.resources.thorlabs.kinesis.benchtop\_stepper.StepParameters method), 467
- wait\_for\_message()** (msl.equipment.resources.thorlabs.kinesis.integrated\_step\_parameters.IntegratedStepParameters method), 490
- wait\_for\_message()** (msl.equipment.resources.thorlabs.kinesis.kcube\_dc\_servo.DC\_ServoParameters method), 523
- wait\_for\_message()** (msl.equipment.resources.thorlabs.kinesis.kcube\_stepper.StepperParameters method), 549
- wait\_for\_message()** (msl.equipment.resources.dataray.datarayocx\_32.DataRayocx32 method), 127
- wait\_for\_message()** (msl.equipment.resources.dataray.datarayocx\_64.DataRayocx64 method), 128
- wait\_until\_ready()** (msl.equipment.resources.picotech.picoscope.picoscope.PicoScope method), 316
- WAITLOCK** (msl.equipment.vxi11.OperationFlag attribute), 590



- attribute), 586
- wavelength (msl.equipment.resources.bentham.benhw64.BenHw64 attribute), 299
- property), 125
- wDigOPs (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 575
- WheelAcceleration (msl.equipment.resources.thorlabs.kinesis.structs.KNAutoVibParams attribute), 567
- WheelAdjustRate (msl.equipment.resources.thorlabs.kinesis.structs.KNAutoVibParams attribute), 571
- WheelDirectionSense (msl.equipment.resources.thorlabs.kinesis.structs.KNAutoVibParams attribute), 567
- WheelMaxVelocity (msl.equipment.resources.thorlabs.kinesis.structs.KMAutoVibParams attribute), 567
- WheelMode (msl.equipment.resources.thorlabs.kinesis.structs.KMAutoVibParams attribute), 567
- WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 275
- WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000AWaveType attribute), 265
- WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 297
- WHITE\_NOISE (msl.equipment.resources.picotech.picoscope.enums.PS5000AWaveType attribute), 288
- WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS3000AExtraOperations attribute), 237
- WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS3000AExtraOperations attribute), 254
- WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations attribute), 269
- WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations attribute), 264
- WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS5000AExtraOperations attribute), 293
- WHITENOISE (msl.equipment.resources.picotech.picoscope.enums.PS5000AExtraOperations attribute), 306
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_ThresholdMode attribute), 239
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS2000A\_ThresholdMode attribute), 229
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_ThresholdMode attribute), 256
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS3000A\_ThresholdMode attribute), 246
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_ThresholdMode attribute), 278
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS4000A\_ThresholdMode attribute), 266
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_ThresholdMode attribute), 299
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_ThresholdMode attribute), 308
- WINDOW (msl.equipment.resources.picotech.picoscope.enums.PS5000A\_ThresholdMode attribute), 100
- wReserved (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 575
- wReserved (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 575
- WRITE (msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations attribute), 276
- write() (msl.equipment.connection\_message\_based.ConnectionMessageBasedClient method), 70
- write() (msl.equipment.connection\_prologix.ConnectionPrologixClient method), 70
- write() (msl.equipment.hislip.HiSLIPClient method), 70
- write() (msl.equipment.vxi11.RPCClient method), 70
- write\_authentication\_exchange() (msl.equipment.vxi11.RPCClient method), 70
- write\_termination (msl.equipment.connection\_message\_based.ConnectionMessageBasedClient property), 28
- write\_termination (msl.equipment.connection\_prologix.ConnectionPrologixClient property), 28
- x (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 574
- xFeedbackSignedGain (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 574
- XML (msl.equipment.resources.picotech.picoscope.enums.PS4000AExtraOperations attribute), 276
- xml\_comment() (in module msl.equipment.utils), 581
- xml\_element() (in module msl.equipment.utils), 581
- Y (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 574
- Year (msl.equipment.resources.nkt.nktpdll.DateTimeType attribute), 153
- yFeedbackSignedGain (msl.equipment.resources.thorlabs.kinesis.structs.QD\_Position attribute), 574

*attribute*), [574](#)

## Z

`zero_calibration()`

(*msl.equipment.resources.bentham.benhw32.Bentham32*  
*method*), [123](#)

`zero_calibration()`

(*msl.equipment.resources.bentham.benhw64.Bentham*  
*method*), [125](#)

`zero_voltage_monitor()`

(*msl.equipment.resources.optronic\_laboratories.ol\_current\_source.OLCurrentSource*  
*method*), [219](#)

`ZFS206` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZFS206` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZFS213` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZFS213` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZFS225` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZFS225` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZMQ` (*msl.equipment.constants.Interface attribute*),  
[44](#)

`ZST13` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZST13` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZST206` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZST206` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZST213` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZST213` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZST225` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZST225` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZST25` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZST25` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)

`ZST6` (*msl.equipment.resources.thorlabs.kinesis.enums.KST\_Stages*  
*attribute*), [457](#)

`ZST6` (*msl.equipment.resources.thorlabs.kinesis.enums.TST\_Stages*  
*attribute*), [461](#)