

---

# PyTek Documentation

*Release 1.1 (v1.1.1.0-r5)*

**Brian Mearns**

April 14, 2014







**PyTek** provides a python API for interacting with Tektronix oscilloscopes over a serial interface. It currently supports some basic commands for the TDS3k series of DPO's (Digital Phosphor Oscilloscopes), especially `capturing` waveforms and `screen shots` from the device.

---

**Note: Serial Port not Included**

PyTek relies on a thirdparty serial port for communications, specifically one that matches the `pyserial` API. It is recommended that you simply use `pyserial` itself.

---



---

## Getting Started

---

To get started, try the *README*, or for complete documentation, check out the *pytek module* API documentation page.





---

## Documentation Contents:

---

### 2.1 README

**PyTek** provides a python API for interacting with Tektronix oscilloscopes over a serial interface. It currently supports some basic commands for the TDS3000 series of Digital Phosphor Oscilloscopes, especially *capturing waveforms* and *screen shots* from the device.

---

**Note: Serial Port not Included**

PyTek relies on a thirdparty serial port for communications, specifically one that matches the [pyserial](#) API. It is recommended that you simply use [pyserial](#) itself.

---

#### Page Contents

- tl;dr
  - What?
  - Install?
  - Serial?
  - Examples?
  - Dependencies?
  - Extras?
  - Docs?
- Misc.
  - Contact Information
  - Copyright and License

#### 2.1.1 tl;dr

##### What?

A python package that gives you an API for interacting with supported Tektronix oscilloscopes over a serial interace.

##### Install?

```
$ pip install pytek
```

Or, from source:

```
$ python setup.py install
```

## Serial?

We don't provide a serial port implementation. We suggest, `pyserial`:

```
$ pip install pyserial
```

## Examples?

```
>>> from serial import Serial
>>> from pytek import TDS3k
>>>
>>> port = Serial("COM1", 9600, timeout=1)
>>> tds = TDS3k(port)
>>>
>>>
>>> # Make the scope identify itself.
...
>>> tds.identify()
'TEKTRONIX,TDS 3034,0,CF:91.1CT FV:v2.11 TDS3GM:v1.00 TDS3FFT:v1.00 TDS3TRG:v1.00'
>>>
>>>
>>> # Capture waveform data
...
>>> waveform = tds.get_waveform(start=100, stop=109)
>>> waveform
<generator object <genexpr> at 0x0238B8A0>
>>
>>> for x,y in waveform:
...     print x, y
...
-0.0045 -0.16
-0.004499 -0.04
-0.004498 -0.04
-0.004497 -0.12
-0.004496 -0.12
-0.004495 -0.08
-0.004494 -0.12
-0.004493 -0.16
-0.004492 -0.2
-0.004491 -0.08
>>>
>>> tds.x_units()
's'
>>> tds.y_units()
'V'
>>>
>>>
>>> # Grab a screen shot (this will take a few minutes).
...
>>> ofile = open("screenshot.tiff", "wb")
```

```
>>> tds.screenshot(ofile, "tiff")
>>>
>>>
>>>
>>> #Fin.
...
>>> tds.close()
>>>
```

## Dependencies?

You'll need a serial port interface. See the "Serial?" section, above.

To build the sphinx docs from source (as is), you'll need the **'sphinx\_rtd\_theme'**:

```
$ pip install sphinx_rtd_theme
```

## Extras?

PyTek package includes the following extras (optional installs):

**serial** Adds `pyserial` package as a requirement, the recommended serial port interface.

**docs** Adds **'sphinx\_rtd\_theme'** package as a requirement, needed for building sphinx docs.

## Docs?

- [Read The Docs \(.org\)](#)
- [Python Hosted \(.org\)](#)

## 2.1.2 Misc.

### Contact Information

This project is currently hosted on [bitbucket](#), at <https://bitbucket.org/bmearns/pytek/>. The primary author is Brian Mearns: you can contact Brian through bitbucket at <https://bitbucket.org/bmearns>.

## Copyright and License

PyTek is *free software*: you can redistribute it and/or modify it under the terms of the **GNU General Public License** as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

PyTek is distributed in the hope that it will be useful, but **without any warranty**; without even the implied warranty of *merchantability* or *fitness for a particular purpose*. See the GNU General Public License for more details.

A copy of the GNU General Public License is available in the PyTek distribution under the file LICENSE.txt. If you did not receive a copy of this file, see <http://www.gnu.org/licenses/>.

## 2.2 pytek module

This is the top level module of the *PyTek* package. It provides classes for interfacing with various Tektronix oscilloscopes over a serial interface.

Most classes in this module are based on a specific series of devices, based on the serial interface supported by the devices. There is currently only one class provided, TDS3k which supports the TDS 3000 series of devices.

---

### Note: Serial Port not Included

pytek relies on a thirdparty serial port for communications, specifically one that matches the [pyserial](#) API. It is recommended that you simply use pyserial itself.

---

**class** pytek.TDS3k(*port*)

Bases: pytek.util.Configurable

The TDS3k class provides functions for interacting with the TDS 3000 series of DPO's from Tektronix. Documentation on this interface is available from Tektronix at [this link](#).

Instances of this class are instantiated by passing in a serial port object, which supports the [pyserial](#) interface. This is the port that the object will use for interacting with the device. Configuration of this port depends on your device and your serial port implementation. Typical settings for RS232 are 9600 baud.

Example:

```
#Import class
from pytek import TDS3k

#Import pyserial
import serial

port = serial.Serial("COM1", 9600, timeout=1)
tds = TDS3k(port)

# ... do stuff with the tds object.

#Closes the object's port.
tds.close()
```

#### Warning: Serial Port Timeout

It is **very important** that you specify a timeout on your serial port. The `get_response` method (used by things like `screenshot` and `get_curve`) continue to read data until a read timeout, so if there is no timeout, it will never return.

**ID\_REGEX** = <\_sre.SRE\_Pattern object at 0x185ae80>

The regular expression used to match the start of the `identify` string, for `sanity_check`.

```
r'^TEKTRONIX,TDS 3\d{3},'
```

**close()**

Closes the object's port by invoking it's `close` method.

The object itself is not affected by this so if you call any methods that try to communicate over the port, it will be trying to communicate over a closed port.

**send\_command** (*command*[, *arg1*[, *arg2*[, ... ] ] ])

Sends a command and any number of arguments to the device. Does not wait for response.

**See also:**

- `send_query` - To send a query and get a one-line response.

**send\_query(query)**

Sends a query to the device and reads back one line, returning that line (stripped of trailing whitespace).

A '?' and a linebreak are automatically appended to the end of what you send.

E.g.:

```
>>> tek.send_query("*IDN")
'TEKTRONIX,TDS 3034,0,CF:91.1CT FV:v2.11 TDS3GM:v1.00 TDS3FFT:v1.00 TDS3TRG:v1.00'
>>>
```

**Warning:** This method turns off header echoing from the device. I.e., it sends "HEADER OFF" before anything else (through the `headers_off` method). If you're expecting headers to be on subsequently, you will need to turn them on with "HEADER ON", or with the `headers_on` method.

**query\_quoted\_string(query)**

Like `send_query`, but expects a quoted string as a response, and strips the quotes off the response before returning. Raises a `ValueError` if the response is not quoted.

**get\_response()**

Simply reads data from the object's `port`, one byte at a time until the port timesout on read. Returns the data as a `str`.

Waits indefinitely for the first byte.

**headers\_off()**

Sends the "HEADER OFF" command to the device, to disable echoing of headers (command names) in query responses from the device. Most methods that query the device will cause this to be sent. You can turn it back on with `headers_on`, or by sending the "HEADER ON" command.

**headers\_on()**

Sends the "HEADER ON" command to the device. See `headers_off` for details.

**identify()**

Convenience function for sending the "\*IDN" query, with `send_query`, and returning the response from the device. This provides information about the device including model number, options, application modules, and firmware version.

**See also:**

- `sanity_check` uses the response from this method to determine if the connected device appears to a supported model.

**sanity\_check()**

Does a sanity check on the device to make sure that the way it identifies itself matches the expected response. Returns `True` if the sanity check passes, otherwise `False`.

The device does not actually enforce this test, and will not perform it automatically (i.e., only if you call this method). This is for your sake so you don't waste time on a device that isn't compatible.

**See also:**

- `identify`
- `force_sanity`

**force\_sanity()**

Does the `sanity_check` on the device, and raises an `Exception` if the check fails.

**acquire\_state** ([*val*])

Configures or queries the value of the `ACQUIRE:STATE` setting on the device. If a value is given, then the setting is configured to the given value. If the value is `None` (the default), then the setting is queried and the value is returned.

For *queries*, return `True` or `False`:

- `True` if the device replies with any of the following: "1", "ON", "RUN"
- `False` otherwise.

For *configuring*, if *val* evaluates as `True`, causes "1" to be sent to the device. Any other value for *val* causes "0" to be sent.

The `ACQUIRE:STATE` setting is related to the “RUN / STOP” button on the device, and it basically configures whether the device is actually acquiring data or not.

**acquire\_single** ([*val*])

Configures or queries the value of the `ACQUIRE:STOPAFTER` setting on the device. If a value is given, then the setting is configured to the given value. If the value is `None` (the default), then the setting is queried and the value is returned.

For *queries*, return `True` or `False`:

- `True` if the device replies with any of the following: "SEQ", "SEQUENCE"
- `False` otherwise.

For *configuring*, if *val* evaluates as `True`, causes "SEQ" to be sent to the device. Any other value for *val* causes "RUN" to be sent.

The `ACQUIRE:STOPAFTER` setting is related to the “single sequence” button on the device. If `True`, then when the device is set to acquire (e.g., by passing `True` to `acquire_state`), it will only acquire a single sequence, and then stop automatically. Otherwise, it will continue to acquire until it is stopped.

**trigger** ()

Force the device to trigger, assuming it is in `READY` state (see `trigger_state`).

This sends the `TRIGGER FORCE` command to the device.

**trigger\_auto** ([*val*])

The `TRIGGER:A:MODE` is related to the “AUTO” and “NORMAL” selections in the Trigger menu. If set to `True`, the trigger is in “AUTO (Untriggered roll)” mode, in which the device automatically generates a trigger if none is detected.

Otherwise, the device is in “NORMAL” mode, in which the device waits for a valid trigger.

**val = 'trigger'**

**trigger\_state** ()

Returns a string indicating the current trigger state of the device. This queries the `TRIGGER:STATE` setting on the device.

The following list gives the possible return values:

- **auto** - indicates that the oscilloscope is in auto mode and acquires data even in the absence of a trigger (see `trigger_auto`).
- **armed** - indicates that the oscilloscope is acquiring pretrigger information. All triggers are ignored in this state.
- **ready** - indicates that all pretrigger information has been acquired and the oscilloscope is waiting for a trigger.
- **save** - indicates that acquisition is stopped or that all channels are off.

- **trigger** - indicates that the oscilloscope has seen a trigger and is acquiring the posttrigger information.

### **get\_waveform\_preamble()**

Queries the waveform preamble from the device, which details how a waveform or curve will be transferred from the device based on the current settings (as with `get_curve` or `get_waveform`, though note that both of those functions alter settings based on provided parameters, before retrieving the data).

Returns a dictionary of preamble values.

Example:

```
>>> wfm_preamble = tds.get_waveform_preamble()
>>> for k, v in wfm_preamble.iteritems():
...     print k, ":", repr(v)
...
byte_order : 'MSB'
binary_format : 'RP'
x_incr : 1e-06
y_scale : 0.08
number_of_points : 10000
y_unit : '"V"'
encoding : 'BIN'
y_zero : 0.0
point_format : 'Y'
waveform_id : '"Ch1, DC coupling, 2.0E0 V/div, 1.0E-3 s/div, 10000 points, Sample mode"'
x_units : '"s"'
y_offset : 128.0
bits_per_sample : 8
bytes_per_sample : 1
pt_offset : 0
xzero : -0.0045
>>>
```

### **get\_curve(source='CH1', double=True, start=1, stop=10000, preamble=False, timing=False)**

Queries a curve (waveform) from the device and returns it as a set of data points. Note that the points are simply unsigned integers over a fixed range (depending on the `double` parameter), they are not voltage values or similar. Use `get_waveform` to get scaled values in the proper units.

**Warning:** Note that this method will set waveform preamble and data parameters on the device, which have a persistent effect which could alter the behavior of future commands.

If `preamble` or `timing` are `True`, returns a tuple: (`preamble_data`, `data`, `timing_data`), where the `preamble_data` and `timing_data` are only present if the corresponding flag is set.

If neither `preamble` nor `timing` is `True`, then just returns `data` as the sole argument (i.e., `data`, not (`data`,)).

In either case, `data` will be a sequence of data points for the curve. If the `double` parameter is `True` (the default), data points are each double-byte wide, in the range from 0 through 65535 (inclusive). This gives you maximum resolution on your data, but takes longer to transfer. Also note that the device does not necessarily have 16 bits of precision in measurement, but data will be left-aligned to the most significant bits.

If `double` is `False`, then the data points are single-byte each, in the range from 0 through 255 (inclusive).

Regardless of `double`, the minimum value corresponds to one vertical division *below* the bottom of the screen, and the maximum value corresponds to one vertical division *above* the top of the screen.

### **Parameters**

- **source** (*str*) – Optional, specify the channel to copy the waveform from. Default is "CH1".

- **double** (*bool*) – Optional, if `True` (the default), data points are transferred 16-bits per point, otherwise they are transferred 8-bits per point, which may cut off least significant bits but will transfer faster.
- **start** (*int*) – Optional, the data point to start at. The waveform contains up to 10,000 data points, the first point is 1. The default value is 1. If you set this param to `None`, it has the same effect as a 1.
- **stop** (*int*) – Optional, the data point to stop at. See `start` for details. The default value is 10,000 to transfer the entire waveform. If you set this to `None`, it has the same effect as 10,000.
- **preamble** (*bool*) – Controls whether or not the curve's preamble is included in the return value. The curve's preamble is not the same as the waveform preamble that configures the data. The curve's preamble is a string that is transmitted prior to the curve's data points. I'm honestly not sure what it is, but it contains a number which seems to increase with the number of data points transferred.
- **timing** (*bool*) – Controls whether or not timing information is included in the return value. Timing gives the number of seconds it took to transfer the data, as a floating point value.

**get\_waveform** (*source='CH1', double=True, start=1, stop=10000, preamble=False, timing=False*)

Similar to `get_curve`, but uses waveform preamble data to properly scale the received data.

If `preamble` or `timing` are `True`, returns a tuple: (`preamble_data`, `data`, `timing_data`), where the `preamble_data` and `timing_data` are only present if the corresponding flag is set.

If neither `preamble` nor `timing` is `True`, then just returns `data` as the sole argument (i.e., `data`, not (`data`,)).

`data` is a sequence of two tuples, giving the X and Y value for each point, in order across the X-axis from left to right. These are properly scaled based on the waveform settings, Giving, for instance, a value in Volts versus Seconds. Check `x_units` and `y_units` to get the actual units.

**get\_num\_points** ()

Queries the number of points that will be sent in a waveform or curve query, based on the current settings.

This is relevant to functions like `get_waveform` and `get_curve`, but note that those functions set the `DATA:START` and `DATA:STOP` configuration options on the device based on provided parameters, thereby effecting the number of points.

**y\_units** ()

Returns a string giving the units of the Y axis based on the current waveform settings.

Example:

```
>>> tds.y_units()
'V'
>>>
```

**x\_units** ()

Returns a string giving the units of the X axis based on the current waveform settings. Possible values include `'s'` for seconds and `'Hz'` for Hertz.

Example:

```
>>> tds.x_units()
's'
>>>
```

**screenshot** (*ofile=None, fmt='RLE', inksaver=True, landscape=False*)

Grabs a hardcopy/screenshot from the device.



If `ofile` is `None` (the default), simply returns the data as a string. Otherwise, it writes the data to the given output stream.

#### Parameters

- **fmt** (*str*) – Optional, specify the format for the image. Valid values will vary by device, but will be a subset of those listed below. The default is “RLE” which gives a Windows Bitmap file.
- **inksaver** (*bool*) – Optional, if `True` (the default), puts the device into hardcopy-inksaver mode, in which the background of the graticular is white, instead of black. If `False`, sets the device to not be in inksaver mode.
- **landscape** (*bool*) – Optional, if `False` (the default), the image will be in portrait mode, which is probably what you want. If `True`, it will be in landscape mode, which generally means the image will be rotated 90 degrees.

#### Possible supported formats:

The following is a list of the formats that may be supported, but individual devices will only support a subset of these. To see if your device supports a format, use `check_img_format`.

- **TDS3PRT** - For the TDS3000B series only, sets format for the TDS3PRT plug-in thermal printer.
- **BMP** - Grayscale bitmap. This is uncompressed, and very large and slow to transfer.
- **BMPColor** - Colored bitmap. Uncompressed, very large and slow to transfer.
- **DESKJET** - For the TDS3000B and TDS3000C series only, formatted for HP monochrome inkjet printers.
- **DESKJETC** - For the TDS3000B and TDS3000C series only, formatted for HP *color* inkjet printers.
- **EPSColor** - Colored Encapsulated PostScript.
- **EPSMono** - Monochrome Encapsulated PostScript.
- **EPSON** - For the TDS3000B and TDS3000C series only, supports Epson 9-pin and 24-pin dot matrix printers.
- **INTERLEAF** - Interleaf image object format.
- **LASERJET** - For the TDS3000B and TDS3000C series only, supports HP monochrome laser printers.
- **PCX** - PC Paintbrush monochrome image format.
- **PCXcolor** - PC Paintbrush color image format.
- **RLE** - Colored Windows bitmap (uses run length encoding for smaller file and faster transfer).
- **THINKJET** - For the TDS3000B and TDS3000C series only, supports HP monochrome inkjet printers.
- **TIFF** - Tag Image File Format.
- **DPU3445** - Seiko DPU-3445 thermal printer format.
- **BJC80** - For the TDS3000B and TDS3000C series only, supports Canon BJC-50 and BJC-80 color printers.
- **PNG** - Portable Network Graphics.

---

**Note:** The fastest transfer seems to be **RLE**, with **TIFF** close behind (transfer times are less than one minute at 9600 baud). **BMP** and **BMPColor** take a very long time (more than five minutes at 9600 baud).

---

**check\_img\_format** (*fmt*)

Tests if a hardcopy image format is supported by the device. This simply sets the `HARDCOPY:FORMAT` configuration value to the given format, and checks to see if it comes back as the same format.

Return `True` if the format is supported, `False` otherwise.

Resets the `HARDCOPY:FORMAT` back to where it was before returning.

**See also:**

screenshot

**k** = 'trig'

**seq** = ['trigger', 'trig']

`pytek.TDS3xxx`

alias of `TDS3k`

## 2.3 pytek.util module

**class** `pytek.util.Configurator` (*name, get=None, set=None, doc=None*)

Bases: `object`

The `Configurator` class creates helper objects that can be used to easily add methods to a class to configure and query a particular setting on the device.

The easiest way to understand it is by example. First, a stripped down usage example:

```
class MyDevice(object):

    __metaclass__ = Configurator.ConfigurableMeta

    @Configurator.config("FOO:BAR")
    def foobar(self, val):
        return val.lower()

    @foobar.setter
    def foobar(self, val):
        return val.upper()

    @Configurator.config
    def frobbbed(self, val):
        return (val == "ON")

    @frobbbed.setter
    def frobbbed(self, val):
        return "ON" if val else "OFF"
```

And now, a more thorough example, expanded from this:

```
class MyDevice(object):

    #Make sure it uses the ConfigurableMeta class as its metaclass,
    # so Configurator objects in the class definition get replaced with
    # appropriate methods.
    __metaclass__ = Configurator.ConfigurableMeta

    #Just some ordinary instance attributes, which we will be the target of
```

```

# our setting configuring and querying.
__foobar = "TAZ"
__frobbed = "OFF"

#This is where the class actually implements sending command and queries.
# The Configurator objects will call these methods.

def send_command(self, name, arg):
    print "~~~> %s %s" % (name, arg)
    if name == "FOO:BAR":
        if not isinstance(arg, str):
            raise TypeError()
        if arg != arg.upper():
            raise ValueError()
        self.__foobar = arg

    elif name == "FROBBED":
        if arg not in ("ON", "OFF"):
            raise ValueError()
        self.__frobbed = arg

    else:
        raise KeyError()

def send_query(self, name):
    print "??? %s" % name
    if name == "FOO:BAR":
        val = self.__foobar
    elif name == "FROBBED":
        val = self.__frobbed
    else:
        raise KeyError()
    print "    <<<< %s" % val
    return val

#Now, define Configurators for each of our configurable settings.

#First, for the FOO:BAR setting, which will be accessed through a
# function called 'foobar'.

@Configurator.config("FOO:BAR")
def foobar(self, val):
    #Translate a value returned by 'send_query' into a value to return
    # to the calling code.
    return val.lower()

@foobar.setter
def foobar(self, val):
    #Translate a value provided by the calling code into a value that
    # will be passed to 'send_command'.
    return val.upper()

#Now, the FROBBED setting. We can use implicit named in the decorator
# for this one.

```

```
@Configurator.config
def frobbed(self, val):
    """
    +++
    Querying returns True for "ON", and False for "OFF".
    """
    if val == "ON":
        return True
    if val == "OFF":
        return False
    raise ValueError(val)

@frobbed.setter
def frobbed(self, val):
    """
    +++
    Valid values for configuring are True and False, or synonymously
    "ON" and "OFF".
    """
    if val is True or val == "ON":
        return "ON"
    elif val is False or val == "OFF":
        return "OFF"
    raise ValueError()
```

With the above code, you could then do the following:

```
>>> dev = MyDevice()
>>> dev.foobar()
???? FOO:BAR
<<<< TAZ
'taz'
>>>
>>> dev.foobar('razzle-dazzle')
~~~> FOO:BAR RAZZLE-DAZZLE
>>>
>>> dev.foobar()
???? FOO:BAR
<<<< RAZZLE-DAZZLE
'razzle-dazzle'
>>>
>>>
>>> dev.frobbed()
???? FROBBED
<<<< OFF
False
>>> dev.frobbed(True)
~~~> FROBBED ON
>>> dev.frobbed()
???? FROBBED
<<<< ON
True
>>>
>>> dev.frobbed(False)
~~~> FROBBED OFF
>>> dev.frobbed()
???? FROBBED
<<<< OFF
```

```

False
>>>
>>> dev.frobbed("ON")
~~~> FROBBED ON
>>> dev.frobbed()
???? FROBBED
<<<< ON
True
>>>
>>> dev.frobbed("???)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "src\pytek\util.py", line 125, in config
    return self(device, val)
  File "src\pytek\util.py", line 116, in __call__
    self.configure(device, self.name, self.set(device, val))
  File "temp.py", line 94, in frobbed
    raise ValueError()
ValueError
>>>
>>>
>>> help(dev.foobar)
Help on method foobar in module pytek.util:

foobar(device, val=None) method of temp.MyDevice instance
    Configures or queries the value of the ``FOO:BAR`` setting on the device.
    If a value is given, then the setting is configured to the given value.
    If the value is 'None' (the default), then the setting is queried and the value
    is returned.

>>>
>>> help(dev.frobbed)
Help on method frobbed in module pytek.util:

frobbed(device, val=None) method of temp.MyDevice instance
    Configures or queries the value of the ``FROBBED`` setting on the device.
    If a value is given, then the setting is configured to the given value.
    If the value is 'None' (the default), then the setting is queried and the value
    is returned.

    Querying returns True for "ON", and False for "OFF".

    Valid values for configuring are True and False, or synonymously
    "ON" and "OFF".

>>>
>>>

```

### Parameters

- **name** – Specifies the name of the setting accessed by this object. Should be either a callable object with a `__name__` attribute, or a string. Strings will be used directly, callables will be filtered through `func_to_name`.
- **get** (*callable*) – Optional: if given, passed to getter.
- **set** (*callable*) – Optional: if given, passed to setter.

- **doc** (*callable*) – Optional: if given, used as the value of the `doc` attribute.

**DEFAULT\_DOCTSTR** = ‘\nConfigures or queries the value of the “%(NAME)s” setting on the device.\nIf a value is given,

A string used for default value of the `doc` attribute.

**classmethod `configure`** (*device, name, val*)

The final method in this object used to configure the setting, given the raw value to be sent to the device.

This is called by the `__call__` method when appropriate.

This delegates to the `send_command` method of the given device.

#### Parameters

- **device** – The object on which the `send_command` will be invoked.
- **name** (*str*) – The name of the setting, usually the value of the `name` attribute. This is the first arguments passed to `send_command`.
- **val** (*str*) – The raw value to configure the setting to. This is the second argument passed to `send_command`.

**classmethod `query`** (*device, name*)

The final method in this object used to query the setting, returning the raw value from the device. This is called by the `__call__` method when appropriate.

This delegates to the `send_query` method of the given device.

#### Parameters

- **device** – The object on which the `send_query` will be invoked.
- **name** (*str*) – The name of the setting, usually the value of the `name` attribute. This is the only arguments passed to `send_query`.

**create\_method** (*name*)

Creates a method with the given name which can be installed in a class to delegate to this object’s `__call__` method. Sets the name of the method to `name`, and sets the `docstr` (`__doc__`) to the value of this object’s `doc` attribute.

This is used by `ConfigurableMeta` to replace `Configurator` instances in the classes dictionary with functions.

**update\_doc** (*func*)

If the given function has a `docstrig` (`__doc__`), then this object’s `doc` attribute is updated with it. Otherwise, it does nothing.

If `func`’s `docstr` begins with ‘`+++`’ alone on a line (any amount of leading and trailing whitespace), then the remainder of the `docstring` is *appended* to the existing `docstring`, instead of replacing it.

**classmethod `func_to_name`** (*func*)

Derives a setting name from a function. The implementation here just uses the `__name__` attribute of the given `func`, and then uses `str.upper()` to make it all upper case.

This is used in `__init__` if the name is a callable object.

**classmethod `boolean`** (*arg, \*\*kwargs*)

A function decorator utility used to create a `Configurator` object which handles boolean settings. This ends up delegating to `set_boolean` to actually set up the `get` and `set` filters based on responses from the decorated function. All keyword arguments passed to this function are forwarded to `set_boolean`.

Similar to `config`, you can invoke this with *implicit arguments* or *explicit arguments*

For **implicit arguments**, you use this method as a function decorator directly, and the `name` to use is derived from the decorated function with `func_to_name`. In this mode, you can't specify any additional arguments to pass to `set_boolean`.

For **explicit arguments**, you invoke this method directly, and it returns a function decorator. This allows you to pass in a string as the first argument to specify the `name` to use, as well as additional keyword arguments to be forwarded on to `set_boolean`.

#### **classmethod config** (*arg*)

A function decorator utility used to create a `Configurator` object and a function decorator to configure its `getter`.

There are two way to invoke this, using *implicit naming* or *explicit naming*.

For **implicit naming**\*, simply pass a function in directly, or use this function directly as a decorator. For instance:

```
@Configurator.config
def foobar(self, val):
    return val
```

The above code will create a new instance of `cls` (i.e., a `Configurator` object), and will pass the given function `foobar` in as the `name` parameter to the constructor. This in turn will use `func_to_name` to derive a value for the instance's `name` attribute from the function, by default (i.e., in the base `Configurator` class), this is just the name of the function in all uppercase.

The function will also be passed to the instance's `getter` method so that the `foobar` function becomes the instance's `get` filter.

This method will then return the `Configurator` object itself, *not* the wrapped function.

The alternative is **explicit naming**, in which this function is not used *as* a function wrapper, but invoked *to return* a function wrapper. This gives you some added flexibility such as explicitly giving the `name` to use for the `Configurator` object. Otherwise, the behavior is essentially the same.

For instance:

```
@Configurator.config('BAZ:RUFFLE')
def foobar(self, val):
    return val
```

In this case, even though the wrapped function has the same name, "`foobar`", the created `Configurator` object will have a name of "`BAZ:RUFFLE`". Other than that, the effects are the same.

In either case, when code like this appears in a class definition, it means that class will have an attribute named `foobar` whose value is a `Configurator` object. If this class is using the `ConfigurableMeta` metaclass, then this attribute will be replaced by a proper method generated by the `Configurator`'s `create_method` method.

Also note that when the wrapped function is passed to the `Configurator`'s `getter` method, this method will also pass it to `update_doc`, so if the wrapped function has a docstring, the `Configurator` object's `doc` attribute will be set accordingly. When the `ConfigurableMeta` gets a hold of it, the corresponding method it adds to the class will receive this `docstr` from the `Configurator` object.

Note that for the remainder of the class definition, you can use the generated `Configurator` object. For instance, you can follow up either of the above examples with the following:

```
@foobar.setter
def foobar(self, val):
    if val is False:
        return "OFF"
    return "ON"
```

Since at this point the `foobar` symbol is actually a Configurator object, you can use its other decorators such as `setter` and `getter`.

**setter** (*func*)

A function wrapper which sets this object's `set` attribute to the given function and passes the function to `update_doc`, then returns `self`.

The given function should take two arguments and return a string. The first argument will be the device on which the `send_command` method is invoked, the second argument will be the client supplied value they want to configure the setting to. The function should return a corresponding string which will actually be sent to the device.

**getter** (*func*)

Like `setter`, but sets the object's `get` attribute, used for querying the setting from the device.

This is a function wrapper which sets this object's `get` attribute to the given function and passes the function to `update_doc`, then returns `self`.

The given function should take two arguments and return a string. The first argument will be the device on which the `send_query` method is invoked, the second argument will be the value returned from the device by `send_query`. The function should return a corresponding value which will be returned to the user to reflect the string returned by the device.

**set\_boolean** (*func*, *strict=False*, *default=False*, *nocase=False*)

Configures the objects `set` and `get` filters based on a boolean setting.

A boolean setting means the setting has a set of possible values that are partitioned into two subsets: true values and false values. On the python side, any value in these subsets corresponds to a value of `True` or `False`, respectively.

This method sets up the object to filter values accordingly, so that querying the setting always returns `True` or `False`, and configuring the setting can be done with `True` or `False`.

To do so, you have to pass in a function which can be evaluated immediately to get the set of true values and the set of false values. The function should take a single boolean argument, if the argument value is `True`, return the set of true values, otherwise, return the set of false values. The method will then create appropriate set and get filters based on these values and the other parameters passed into this function (see below).

The sets of true values and false values returned by `func` must be sequences. The first value in each sequence will be used as the *canonical* value, meaning the ones that will actually be passed to the device for the corresponding value. All other values in the sets will be acceptable responses from the device for queries, and will result in the corresponding boolean value being returned to the caller.

**See also:**

`boolean`

**Parameters**

- **func** (*callable*) – This function will be called twice, immediately. Once with a value of `True`, which should return a sequence of true values; and once with a value of `False`, which should return a sequence of false values.
- **strict** (*bool*) – Optional, default is `False`. If `True`, then the generated `set` and `get` filters will be strict about values. The `set` filter will only accept boolean values, and will raise a `TypeError` otherwise. The `get` filter will only accept values from the true- and false- value sets, and will raise a `ValueError` if the device returns anything else.

If the value of the parameter is `False`, the generated functions are not as strict, and will not raise exceptions for unrecognized values (the way it handles unrecognized values



depends on the value of the `default` parameter). For the non-strict `set` filter, values are simply evaluated as bools to choose which value to send.

- **default** (*bool*) – Optional, default value is `False`. This is only used if `strict` is `False`, in which case it determines the *default* value when an unrecognized value is encountered.
- **nocase** (*bool*) – Optional, default value is `False`. If `True`, then values are considered case-insensitive.

#### **class ConfigurableMeta**

Bases: `type`

This is a meta class that can be added to classes to more easily support the use of `Configurator` objects as pseudo-methods.

The meta class extends the `__new__` function to find all instances of `Configurator` in the class's dictionary, and replace it with a method created by the `Configurator`'s `create_method` method.

See the example code in the documentation for `Configurator` for an example.

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

#### **class pytek.util.Configurable**

Bases: `object`

Just a simple base classes that uses `ConfigurableMeta` as the metaclass.

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

## 2.4 pytek.version module

The `version` module provides version numbering for the entire *PyTek* package.

### Page Contents

- Versioning
  - Version Number
    - \* Major Version
    - \* Minor Version
    - \* Patch Version
    - \* Semantic Version
    - \* Compatibility Summary
    - \* Version Tag
    - \* Development code
    - \* Specifying a version number
    - \* Interface Version
  - Release Number
- Module Contents

### 2.4.1 Versioning

The *PyTek* packages uses a five part version number, plus an incremental release number. Either the version number or the release number can be used to identify a released version of the code.

## Version Number

The version number is a four part dotted number, with an optional tag on the end. Formally, a version number looks like:

```
version number ::= <Major>.<minor>[.<patch>[.<semantic>]][-[x-]<tag>]
```

With each new released version of the code, exactly one of the four numbers will increase, and any numbers to its right will reset to 0.

The easiest way to understand version numbers is from the perspective of someone who has written *client code*: i.e., code that makes use of a particular version of the *PyTek* library. From this perspective, the version number indicates whether or not your client code can be expected to work with different versions of *PyTek*.

### Major Version

The `<Major>` component is the **major version number**, and it describes *backward compatibility*. Going to a *newer* version of *PyTek*, your code should continue to work as long as the major version doesn't change.

The major version is changed only when something is removed from the *PyTek* public interface. For instance, if a function is no longer supported, the major version number would have to increase, because client code which relied on that function would no longer work.

The major version number can be accessed through the `MAJOR` member of this module.

### Minor Version

The `<minor>` component is the **minor version number**, and it describes *forward compatibility*: Going to an *older* version of *PyTek*, your code will continue to work as long as the minor version doesn't change. (As before, your code will also work for *newer* versions of *PyTek*, as long as the major version number hasn't changed).

The minor version number is changed only when something is added to the *PyTek* public interface, for instance a new function is added. Such a change maintains *backward compatibility* (as described above), but *loses forward compatibility*, because any client code written against this new version may not work with an older version.

The minor version number can be accessed through the `MINOR` member of this module.

### Patch Version

The `<patch>` component is the **patch number**, and it describes changes that *do not affect compatibility*, either forwards or backwards. Your client code will continue to work with an older or newer version of *PyTek* as long as the major and minor version numbers are the same, regardless of the patch number.

Patch changes are code changes that do not effect the interface, for instance bug-fixes or performance enhancements. (although some bugs effect the interface and may therefore cause a higher version number to change).

The patch number can be accessed through the `PATCH` member of this module.

### Semantic Version

The `<semantic>` component is the **semantic version number**, and it describes changes that do not affect how the code runs at all. This generally means that documentation or other auxiliary files included in the package have changed.

The semantic version number can be accessed through the `SEMANTIC` member of this module.

## Compatibility Summary

The following table summarizes compatibility for a hypothetical client application built against released *PyTek* version *M.n.p.s*:

Component	Compatible (all)	Incompatible (any)
Major	M	!= M
minor	>= n	< n
patch	any	
semantic	any	

## Version Tag

The `<tag>` component is the **version tag**, which is used only for non-released code. The tag has one of the following forms:

```
version tag ::=    << empty >>
                  dev[-<rev>]
                  blood-<branch>[-<rev>]
```

The first form is an empty tag, and is reserved for released (tagged) code only.

The second form, "dev", is for non-released code in the *trunk*. This is the main line of development. Dev code may not be completely functional, and may even break the existing interface.

The third form, "blood-...", is for non-released code on a *branch*. The `<branch>` component of this form should be the name of the branch. This is considered *bleeding-edge* code and may be highly unstable.

The optional `<rev>` component on both the second and third forms can be used to specify a specific revision for committed development code. This must be an globally unambiguous identifier for the revision, for instance the change set id.

## Development code

A non-empty version tag indicates a *development version* of the code. In this case, the four version numbers remain *unchanged* until the code is released (in which case it is no longer development code, and the tag is changed to empty).

In other words, anytime you see a non-empty version tag, the version numbers shown refer to version from which the development code is derived. This is done because it is not generally known until release what the next released version number will be, since it is not known what types of changes will be included in it.

## Specifying a version number

When specifying a version number, the major and minor version numbers should always be included. Additionally, all non-zero version numbers should be included, and any version number to the left of a non-zero version number should be included.

The tag should always be included in the version number, with the indicated hyphen separating the semantic version number and the tag. The only exception is for released code, in which case the tag is empty and should be omitted, along with the joining hyphen.

The optional "x-" shown preceding the tag in the version number is for compatibility with setup-tools so that versions compare correctly.

The above rules will unambiguously describe any released version of the package.

## Interface Version

Because any change to the public interface requires a change to either the major or minor version numbers, the interface can be specified by a shortened two part version:

```
interface version ::= <Major>.<minor>
```

Note that this only applies for released versions: development versions may modify the public interface prior to changing the version numbers.

## Release Number

The release number is a simple integer which increments by one for every public release of the code. It does not convey any information about compatibility with other versions, but it does provide a simple alternative to identifying released versions.

The release number should be written with a leading "r" or "rel". For instance, the first release was "r1".

For release code, the release number may be used in place of the tag in the version number. This is optional because the version number and the release number are synonymous. However, including them both in the version string is a useful way to provide both pieces of information.

This alternative form of the version number is:

```
alt. version number ::= <Major>.<minor>[.<patch>[.<semantic>]]-r<release>
```

### 2.4.2 Module Contents

```
pytek.version.RELEASE = 5
```

The current Release Number.

```
pytek.version.MAJOR = 1
```

The current *major version number*.

```
pytek.version.MINOR = 1
```

The current *minor version number*.

```
pytek.version.PATCH = 1
```

The current *patch version number*.

```
pytek.version.SEMANTIC = 0
```

The current *semantic version number*.

```
pytek.version.TAG = None
```

The current Version Tag.

Tag options are None, "dev", and "blood-"

- None means this is a released/tagged version.
- "dev" means this is a development version from the trunk/mainline.
- "blood-" means it's on a branch. After the dash, fill in the name of the branch.

Dev and blood versions are still numbered for the *previous* version, because we may not know what the next version will be until we're finished.

`pytek.version.COPYRIGHT = 2014`

The copyright year for the *PyTek* code.

`pytek.version.YEAR = 2014`

The year in which the code was released.

**See also:**

- `MONTH`
- `DAY`
- `datestr`

`pytek.version.MONTH = 4`

The month in which the code was released. This is 1 indexed, in [1, 12].

**See also:**

- `YEAR`
- `DAY`
- `datestr`
- `MONTH_NAMES`

`pytek.version.DAY = 13`

The day of the month on which the code was released.

**See also:**

- `YEAR`
- `MONTH`
- `datestr`

`pytek.version.MONTH_NAMES = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']`

A sequence giving the names of months, for use by `datestr`. Standard values are three-letter English-language abbreviations for the months of the Gregorian calendar.

`pytek.version.setuptools_string()`

Returns the version string used by `setuptools`. This takes one of two forms:

```
setuptools_string ::= <Major>.<minor>.<patch>.<semantic>-x-<tag>
                   <Major>.<minor>.<patch>.<semantic>-r<release>
```

The first form is used for development code (i.e., when `TAG` is not `None`), and the second it used for released code.

This is similar to `string`, except for the additional `x-` for development versions, which is used to ensure that `setuptools` sorts versions correctly. (specifically, so that released versions are earlier than development versions which are derived from them).

`pytek.version.tag_name()`

Returns the tag name for the most recent release.

`pytek.version.short_string()`

Returns a string describing the Interface Version (i.e., `<Major>.<minor>`).

`pytek.version.string()`

Like `setuptools_string`, except leaves out the `x-` for development versions.

`pytek.version.datestr()`

Returns a simple string giving the date of release. Format of this string is unspecified, it intended to be human readable, not machine parsed. For machine processing, use the individual variables, as listed below.

See also:

- `YEAR`
- `MONTH`
- `DAY`
- `MONTH_NAMES`

## 2.5 LICENSE (GPLv3)

GNU GENERAL PUBLIC LICENSE  
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>>  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for  
software and other kinds of works.

The licenses for most software and other practical works are designed  
to take away your freedom to share and change the works. By contrast,  
the GNU General Public License is intended to guarantee your freedom to  
share and change all versions of a program--to make sure it remains free  
software for all its users. We, the Free Software Foundation, use the  
GNU General Public License for most of our software; it applies also to  
any other work released this way by its authors. You can apply it to  
your programs, too.

When we speak of free software, we are referring to freedom, not  
price. Our General Public Licenses are designed to make sure that you  
have the freedom to distribute copies of free software (and charge for  
them if you wish), that you receive source code or can get it if you  
want it, that you can change the software or use pieces of it in new  
free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you  
these rights or asking you to surrender the rights. Therefore, you have  
certain responsibilities if you distribute copies of the software, or if  
you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether  
gratis or for a fee, you must pass on to the recipients the same  
freedoms that you received. You must make sure that they, too, receive  
or can get the source code. And you must show them these terms so they  
know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

#### TERMS AND CONDITIONS

##### 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

#### 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

#### 2. Basic Permissions.

All rights granted under this License are granted for the term of



copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

### 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

### 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

### 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified

it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

## 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord

with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates

for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

#### 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further

restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

#### 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

#### 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

#### 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner

consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

#### 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

#### 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

#### 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

#### 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS



### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.



---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*



---

**Version**

---

This documentation is for PyTek 1.1 (v1.1.1.0-r5).



---

## Project Resources

---

- [PyTek project homepage \(bitbucket\)](#)
- [PyTek on pypi](#)
- **Online documentation:**
  - [Read The Docs \(.org\)](#)
  - [Python Hosted \(.org\)](#)





---

## External References

---

- TDS3000, TDS3000B & TDS3000C Series Programmer Manual ([Tektronix.com](http://Tektronix.com))



## p

pytek, ??  
pytek.util, ??  
pytek.version, ??