pyTempico

Release 1.1.1

Tausand Electronics

Mar 08, 2025

CONTENTS:

PyTempico is a Python library developed by Tausand Electronics, with Joan Amaya and David Guzman as the lead developers. It enables seamless interaction with Tempico TP1004 devices, facilitating data communication and control. The library relies on key modules such as hid and pyserial for efficient device connectivity. For more information, visit tausand.com

Release 1.1.1

CHAPTER

ONE

PYTEMPICO PACKAGE

Created on Jan 30 2024

@author: David Guzman at Tausand Electronics
dguzman@tausand.com
https://www.tausand.com

Core class and methods for PyTempico library.

To use with Tausand Electronics' Time-to-Digital Converters (TDCs) of the family *Tausand Tempico*.

1.1 TempicoDevicesSearch class

class pyTempico.core.**TempicoDevicesSearch**

A class for discovering Tempico devices.

This class provides methods to search for Tempico devices in a network or connected system.

findDevices()

Finds and verifies whether a device with the given VID and PID is a Tempico device.

This function takes the Vendor ID (VID) and Product ID (PID) as inputs, converts them to integers, and attempts to open the device using these values. It then checks if the manufacturer and product strings match the expected values for a Tempico device.

Parameters

- **vid_s** (*str*) – The Vendor ID (VID) of the device in string format.
- **pid_s** (*str*) – The Product ID (PID) of the device in string format.

Returns *True* if the device is a Tempico, *False* otherwise.

Return type bool

getVidPid(*vid_pid_information*)

Extracts the Vendor ID (VID) and Product ID (PID) from a string and returns them as a tuple.

This function processes a string that contains the VID and PID information in the format 'VID:PID=xxxx:yyyy'. It splits the string and retrieves the VID and PID values, returning them as a tuple of strings.

Parameters **vid_pid_information** (*str*) – A string containing the VID and PID information.

Returns A tuple containing the VID and PID as strings (vid, pid).

Return type tuple

tryOpenDevices(*vid_s*, *pid_s*)
Finds and verifies whether a device with the given VID and PID is a Tempico device.

This function takes the Vendor ID (VID) and Product ID (PID) as inputs, converts them to integers, and attempts to open the device using these values. It then checks if the manufacturer and product strings match the expected values for a Tempico device.

Parameters

- **vid_s** (*str*) – The Vendor ID (VID) of the device in string format.
- **pid_s** (*str*) – The Product ID (PID) of the device in string format.

Returns *True* if the device is a Tempico, *False* otherwise.

Return type bool

verifyPyTempico(*tuple_vid_pid*)
Verifies whether the connected device is a Tempico device.

This function checks if the device's Vendor ID (VID) and Product ID (PID) match the values corresponding to a Tempico device. It returns *True* if the device is identified as a Tempico, and *False* otherwise.

Parameters **tuple_vid_pid** (*tuple*) – A tuple containing the VID and PID of the device.

Returns *True* if the device is a Tempico, *False* otherwise.

Return type bool

1.2 TempicoDevice class

class pyTempico.core.**TempicoDevice**(*com_port*)
Tausand Tempico TDC device object.

To create an object of the TempicoDevice class, it is required to send as parameter the desired com_port. For example,

```
>>> my_tempico_device_object = pyTempico.TempicoDevice('COM5')
```

To modify or access attributes, **please use methods**. For example,

```
>>> my_tempico_device_object.getIdn()
```

To access attributes of a particular channel, use methods of the TempicoChannel class through attributes ch1, ch2, ch3, ch4 of this class. For example, to get average cycles on channel 2,

```
>>> my_tempico_device_object.ch2.getAverageCycles()
```

Changing attributes without using methods, do not change the actual parameters in the device.

Accesing attributes without using methods, returns values registered in local memory, that may not be updated.

To begin a measurement and read its results, use methods *measure()* and *fetch()*.

id_tempico_device
Unique identifier of the TausandDevice object.

Type int

ch1
Object of the TempicoChannel class linked to TDC in channel 1 (input A).

Type *TempicoChannel*

ch2
Object of the TempicoChannel class linked to TDC in channel 2 (input B).

Type *TempicoChannel*

ch3
Object of the TempicoChannel class linked to TDC in channel 3 (input C).

Type *TempicoChannel*

ch4
Object of the TempicoChannel class linked to TDC in channel 4 (input D).

Type *TempicoChannel*

device
Serial port object.

Type Serial

idn
Identification string.

Type str

number_of_channels
number of stop inputs of the device.

Type int

number_of_runs
Number of measurement runs of the TDCs in TempicoDevice.

Type int

port
Serial port string.

Type str

threshold
Threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

Type float

abort()
Cancels an ongoing measurement on the TempicoDevice.

This function sends a cancel command to the TempicoDevice to stop any measurement currently in progress. It ensures that all measurement processes are halted and the device is ready for a new operation or safely turned off.

This function requires that a connection is established with the TempicoDevice.

Parameters (none) –

close()
Ends (closes) a connection with a TausandDevice.

It is recommended to close connection at the end of a routine, to free the device's port for future use.

Parameters (none) –

closeTempico()
Ends (closes) a connection with a TausandDevice.

Same as method *close()*.

Parameters **(none)** –

convertReadDataToFloatList(*data_string*)
Converts a string with a read dataset message issued by a TempicoDevice, into an float 2D-list.

The dataset of a TempicoDevice is in the following format:

```
[[ch,run,start_s,stop_ps1,...,stop_psN],...,[ch,run,start_time_us,stop_ps1,...,
↪stop_psN]]
```

where

- 'ch' indicates the TDC channel,
- 'run' goes from 1 to NumberOfRuns,
- 'start_s' is the epoch timestamp of start pulse, in seconds, with microseconds precision; this value overflows (go back to zero) after 2^32 seconds,
- 'stop_ps1' is the measured precision timelapse between start and the 1st stop pulse, in picoseconds,
- 'N' is the NumberOfStops.

Every value in the dataset is converted to a float.

Parameters **data_string** (*str*) – dataset message to convert.

Returns dataset message converted.

Return type list(float)

convertReadDataToNumberList(*data_string*)
Converts a string with a read dataset message issued by a TempicoDevice, into an number of number 2D-list (integer or float).

The dataset of a TempicoDevice is in the following format:

```
[[ch,run,start_s,stop_ps1,...,stop_psN],...,[ch,run,start_time_us,stop_ps1,...,
↪stop_psN]]
```

where

- 'ch' (int) indicates the TDC channel,
- 'run' (int) goes from 1 to NumberOfRuns,
- 'start_s' (float) is the epoch timestamp of start pulse, in seconds, with microseconds precision; this value overflows (go back to zero) after 2^32 seconds,
- 'stop_ps1' (int) is the measured precision timelapse between start and the 1st stop pulse, in picoseconds,
- 'N' (int) is the NumberOfStops.

Every value in the dataset is either an integer or a float.

Parameters **data_string** (*str*) – dataset message to convert.

Returns dataset message converted.

Return type list(number)

decrementThresholdVoltage()

Reduces the threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

To validate the actual threshold voltage applied, method *getThresholdVoltage()* should be called.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

fetch()

Reads the most recent measurement data set form a TempicoDevice.

The dataset of a TempicoDevice is in the following format:

```
[[ch,run,start_s,stop_ps1,...,stop_psN],...,[ch,run,start_time_us,stop_ps1,...,
↪stop_psN]]
```

where

- 'ch' (int) indicates the TDC channel,
- 'run' (int) goes from 1 to NumberOfRuns,
- 'start_s' (float) is the epoch timestamp of start pulse, in seconds with microsecond resolution. This value overflows (go back to zero) after 2^32 seconds,
- 'stop_ps1' (int) is the measured precision timelapse between start and the 1st stop pulse, in picoseconds,
- 'N' (int) is the NumberOfStops.

Every value in the dataset is either an integer or a float number.

If no measurement has been done, the device may respond with an empty dataset. To make a new measurement, method *measure()* must be used.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

Returns measured dataset.

Return type list(number)

getBaudRate()

Returns the TempicoDevice baud rate.

Parameters **(none)** –

Returns baud rate.

Return type int

getFirmware()

Returns the TempicoDevice firmware version.

Parameters **(none)** –

Returns firmware version.

Return type str

getIdn()

Returns the TempicoDevice identification string.

If the connection is established with the TempicoDevice, this function request the device for the string. If not, the most recent read string is returned.

Parameters (none) –

Returns identification string.

Return type str

getNumberOfRuns()

Returns the number of measurement runs of the TDCs in TempicoDevice.

By default, number of runs = 1 (single measurement).

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns Number of number of runs.

Return type integer

getSettings()

Reads the current settings form a TempicoDevice.

The response for settings query on a TempicoDevice is in the following format:

```
CH1:ACYC 1;CH1:ENAB 1;CH1:NST 1;...;CH4:STOP:MASK 0;NRUN 1;THR 1.00
```

This function requires that a connection is established with the TempicoDevice.

Parameters (none) –

Returns device settings.

Return type str

getThresholdVoltage()

Returns the threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

Start and stop inputs are coupled to 50 ohms.

By default, threshold voltage = 1.00V (recommended for TTL>2.5V).

All inputs are 5V tolerant.

Gate input. This parameter does not have effect on the gate input. Gate input accepts 3.3V TTL and 5V TTL signals.

- When gate is disconnected, system is enabled.
- When gate is connected to 0V, system is disabled.
- When gate is connected to 3.3V/5V, system is enabled.

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns start and stop inputs threshold voltage.

Return type float

incrementThresholdVoltage()

Increases the threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

To validate the actual threshold voltage applied, method *getThresholdVoltage()* should be called.

This function requires that a connection is established with the TempicoDevice.

Parameters (none) –

`isOpen()`
Returns if a TDC TempicoDevice connection is established (open).

Parameters (none) –

Returns True when TempicoDevice connection is open.

Return type bool

`isPendingReadMessage()`
Determines if a pending message is available to be read in a TempicoDevice serial port.

Parameters (none) –

Returns True, when a pending message is found.

Return type bool

`measure()`
Begins a measurement sequence and reads its dataset from a TempicoDevice.

The dataset of a TempicoDevice is in the following format:

```
[[ch,run,start_s,stop_ps1,...,stop_psN],...,[ch,run,start_time_us,stop_ps1,...,
↪stop_psN]]
```

where

- 'ch' (int) indicates the TDC channel,
- 'run' (int) goes from 1 to NumberOfRuns,
- 'start_s' (float) is the epoch timestamp of start pulse, in seconds with microsecond resolution. This value overflows (go back to zero) after 2^32 seconds,
- 'stop_ps1' (int) is the measured precision timelapse between start and the 1st stop pulse, in picoseconds,
- 'N' (int) is the NumberOfStops.

Every value in the dataset is either an integer or a float number.

If measurement cannot be completed within timeout, the device may respond with an incomplete or empty dataset. In this case, to obtain a complete dataset, the method *fetch()* may be called later.

This function requires that a connection is established with the TempicoDevice.

Parameters (none) –

Returns measured dataset.

Return type list(number)

`open()`
Establishes (opens) a connection with a TausandDevice.

It is mandatory to establish a connection with this method before to be able to send/to receive data to/from the device.

Parameters (none) –

`openTempico()`
Establishes (opens) a connection with a TausandDevice.

Same as method *open()*.

Parameters **(none)** –

readIdnFromDevice()

Returns the TempicoDevice identification string, by requesting it to the device.

This function requires that a connection is established with the TempicoDevice. As an alternative, method *getIdn()* may be used.

Parameters **(none)** –

Returns identification string.

Return type str

readMessage()

Reads pending messages sent by a TempicoDevice from its serial port.

If no message is received, it waits the port timeout, typically 1s.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

Returns read message.

Return type str

reset()

Sends a reset command to the TempicoDevice.

Applying a reset clears all the settings of the TempicoDevice and its TempicoChannels to their default values.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

selfTest()

Performs a self-test on the TempicoDevice hardware.

This function initiates a self-diagnostic test on the TempicoDevice to verify its hardware integrity. If the self-test is successful, it prints the message "Self test passed. Device is working properly." If the self-test fails, it prints the message "Self test failed. Device may have a problem.", indicating a potential issue with the hardware that may require further investigation or support.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

Returns None

setNumberOfRuns(*number*)

Modifies the number of measurement runs of the TDCs in TempicoDevice.

By default, number of runs = 1 (single measurement).

This function requires that a connection is established with the TempicoDevice.

Parameters **number** (*int*) – desired number of runs for every TDC. Valid values are from 1 to 1000.

setThresholdVoltage(*desired_voltage*)

Changes the threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

Start and stop inputs are coupled to 50 ohms.

By default, threshold voltage = 1.00V (recommended for TTL>2.5V).

All inputs are 5V tolerant.

Gate input. This parameter does not have effect on the gate input. Gate input accepts 3.3V TTL and 5V TTL signals.

- When gate is disconnected, system is enabled.
- When gate is connected to 0V, system is disabled.
- When gate is connected to 3.3V/5V, system is enabled.

To validate the actual threshold voltage applied, method *getThresholdVoltage()* should be called.

This function requires that a connection is established with the TempicoDevice.

Parameters **desired_voltage** (*float*) – desired start and stop inputs threshold voltage. Valid parameters are MINimum|MAXimum|DOWN|UP or a number from 0.90 to 1.60.

setThresholdVoltageToMaximum()

Sets to the maximum valid value the threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

To validate the actual threshold voltage applied, method *getThresholdVoltage()* should be called.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

setThresholdVoltageToMinimum()

Sets to the minimum valid value the threshold voltage on the rising edge of start and stops inputs of TDCs in the TempicoDevice.

To validate the actual threshold voltage applied, method *getThresholdVoltage()* should be called.

This function requires that a connection is established with the TempicoDevice.

Parameters **(none)** –

waitAndReadMessage(*wait_time_ms=1*)

Waits the specified time, and then reads pending messages sent by a TempicoDevice from its serial port, if any.

If no message is received, it does not wait for a port timeout.

This function requires that a connection is established with the TempicoDevice.

Parameters **wait_time_ms** (*int, optional*) – Waiting time, in miliseconds. Defaults to 1.

Returns read message.

Return type str

writeMessage(*message*)

Writes a message to a TempicoDevice in its serial port.

If a response is expected after writing a message, the *readMessage()* method should be called afterwards to obtain the response.

This function requires that a connection is established with the TempicoDevice.

Parameters **message** (*str*) – message to be sent.

1.3 TempicoChannel class

class pyTempico.core.**TempicoChannel**(*id_device*, *ch_num*)

Single channel on a Tempico Device.

To modify or access attributes, **please use methods**. For example, to get average cycles on channel 2,

```
>>> my_tempico_device_object.ch2.getAverageCycles()
```

Changing attributes without using methods, do not change the actual parameters in the device.

Accesing attributes without using methods, returns values registered in local memory that may not be updated.

id_tempico_channel

Unique identifier for a TempicoChannel object.

Type int

id_tempico_device

Identifier of the TausandDevice linked to this TempicoChannel object.

Type int

average_cycles

Average cycles.

Type int

channel_number

Number of the channel in the device (1=A, 2=B,...).

Type int

enable

True when the channel is enabled.

Type bool

mode

Measurement mode. 1|2.

- 1: Short measurement range. Start-stop times from 12ns to 500ns.
- 2: Large measurement range. Start-stop times from 125ns to 4ms.

Type int

number_of_stops

Number of stop pulses expected after a start pulse arrives. 1..5.

Type int

parent_tempico_device

Pointer reference to parent object of TempicoDevice() class.

Type *TempicoDevice*

start_edge

Edge type on the start pulse used to begin timing. RISE|FALL.

Type str

stop_edge

Edge type on the stop pulses used to end timing. RISE|FALL.

Type str

stop_mask
Time that stop pulses are ignored after receiving a start pulse on the TDC. Value in microseconds. 0..4000.

Type int

disableChannel()
Disables a TDC TempicoChannel.

By default, channels are enabled.

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

To validate the status of the channel, method *isEnabled()* may be used.

Parameters (none) –

enableChannel()
Enables a TDC TempicoChannel.

By default, channels are enabled.

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

To validate the status of the channel, method *isEnabled()* may be used.

Parameters (none) –

getAverageCycles()
Returns the average cycles of the TDC TempicoChannel.

By default, average cycles = 1 (no multi-cycle averaging).

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns Number of average cycles.

Return type integer

getMode()
Returns the measurement mode of the TDC TempicoChannel.

By default, mode = 1.

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns

Mode. Possible values are,

- 1: Short measurement range. Start-stop times from 12ns to 500ns.
- 2: Large measurement range. Start-stop times from 125ns to 4ms.

Return type integer

getNumberOfStops()
Returns the expected number of stop pulses of the TDC TempicoChannel.

By default, number of stops = 1 (single start -> single stop).

The TDC must receive all the expected number of stops to register them as a valid measurement; otherwise, the measurements are discarded.

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns Number of stops.

Return type integer

`getStartEdge()`

Returns the edge type used on start pulses of the TDC TempicoChannel.

By default, start edge = 'RISE'.

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns

start edge type. Possible values are,

- 'RISE': TDC timing starts on a rising edge of the start pulse.
- 'FALL': TDC timing starts on a falling edge of the start pulse.

Return type string

`getStopEdge()`

Returns the edge type used on stop pulses of the TDC TempicoChannel.

By default, stop edge = 'RISE'.

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns

stop edge type. Possible values are,

- 'RISE': TDC timing ends on a rising edge of the stop pulse.
- 'FALL': TDC timing ends on a falling edge of the stop pulse.

Return type string

`getStopMask()`

Returns the time that stop pulses are ignored after receiving a start pulse on the TDC TempicoChannel. In microseconds.

By default, stop mask = 0 (no masking).

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters (none) –

Returns stop mask time, in microseconds.

Return type integer

`isEnabled()`
Returns if a TDC TempicoChannel is enabled.

By default, channels are enabled.

If the connection is established with the TempicoDevice, this function request the device for the value. If not, the most recent value is returned.

Parameters `(none)` –

Returns True, when TDC channel is enabled.

Return type bool

`setAverageCycles(number)`
Modifies the average cycles of the TDC TempicoChannel.

By default, average cycles = 1 (no multi-cycle averaging).

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

Parameters `number` (*int*) – desired average cycles for the TDC. Valid values are 1|2|4|8|16|32|64|128.

`setMode(number)`
Modifies the measurement mode of the TDC TempicoChannel.

By default, mode = 1. Possible values are,

- 1: Short measurement range. Start-stop times from 12ns to 500ns.
- 2: Large measurement range. Start-stop times from 125ns to 4ms.

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

Parameters `number` (*int*) – desired measurement mode for the TDC. Valid values are 1 or 2.

`setNumberOfStops(number)`
Modifies the expected number of stop pulses of the TDC TempicoChannel.

By default, number of stops = 1 (single start -> single stop).

The TDC must receive all the expected number of stops to register them as a valid measurement; otherwise, the measurements are discarded. For extending the valid time range, consider using measurement mode 2.

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

Parameters `number` (*int*) – desired number of stops for the TDC. Valid values are from 1 to 5.

`setStartEdge(edge_type)`
Sets the edge type used on start pulses of the TDC TempicoChannel.

By default, start edge = 'RISE'. Possible values are,

- 'RISE': TDC timing starts on a rising edge of the start pulse.
- 'FALL': TDC timing starts on a falling edge of the start pulse.

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

Parameters `edge_type` (*str*) – desired start edge type for the TDC. Valid values are 'RISE', 1, 'FALL', 0.

`setStopEdge(edge_type)`
Sets the edge type used on stop pulses of the TDC TempicoChannel.

By default, stop edge = 'RISE'. Possible values are,

- 'RISE': TDC timing ends on a rising edge of the stop pulse.
- 'FALL': TDC timing ends on a falling edge of the stop pulse.

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

Parameters **edge_type** (*str*) – desired stop edge type for the TDC. Valid values are 'RISE', 1, 'FALL', 0.

setStopMask(*stop_mask_in_us*)

Modifies the time that stop pulses are ignored after receiving a start pulse on the TDC TempicoChannel.

By default, stop mask = 0 (no masking).

This function requires that a connection is established with the TempicoDevice of the TempicoChannel.

Parameters **stop_mask_in_us** (*int*) – desired stop mask for the TDC, in microseconds. Valid values are from 0 to 4000.

CHAPTER

TWO

EXAMPLES

2.1 modeTwoMeasureExample.py

```
# -*- coding: utf-8 -*-
"""modeTwoMeasureExample

    Created on Tue May  7 11:42 2024

    Connects to a Tausand Tempico device, changes settings to measure only in
    channel 1 in mode 2 (125ns-4ms mode). Then, starts a measurement and reads
    the results.

    Instructions:
        * before running this example, pyTempico package must be installed.
        * change 'my_port' to your corresponding port.
        * connect signals to your Tempico Device. If no signals are measured,
        this example will return an empty data array.
        * run.

    | @author: David Guzman at Tausand Electronics
    | dguzman@tausand.com
    | https://www.tausand.com
"""

import pyTempico

my_port = 'COM16' #change this port to your Tempico device's port

my_device = pyTempico.TempicoDevice(my_port)    #create object

print('opening connection with device in port',my_port)
my_device.open()             #open connection with device
if my_device.isOpen():
    print('connection with my_device is open')
else:
    print('connection with my_device is close')

print('\nreseting device. This clears previous measurements, and changes settings to␣
↪default values.')
my_device.reset()
```

(continues on next page)

(continued from previous page)

```
#Enable channel 1, disable channels 2-4
print('\ndisabling channels 2-4.')
my_device.ch1.enableChannel()   #optional, since enabled by default
my_device.ch2.disableChannel()
my_device.ch3.disableChannel()
my_device.ch4.disableChannel()

print('\nchanging measurement mode in channel 1 to mode 2')
my_device.ch1.setMode(2)    #default mode is 1, changing to mode 2
#verify
print('my_device.ch1.getMode():',my_device.ch1.getMode())

print('\nsending a measure request to device')
my_device.measure()   #starts a measurement, and saves response in 'data'
data = my_device.measure()   #starts a measurement, and saves response in 'data'
print('measured data, in ps:',data)

print('fetch:',my_device.fetch()) #fetch most recent data

my_device.close()            #close connection with device
if my_device.isOpen():
    print('connection with my_device is open')
else:
    print('connection with my_device is close')
```

2.2 readSettingsExample.py

```
# -*- coding: utf-8 -*-
"""readSettingsExample

    Created on Tue May  7 09:49 2024

    Connects to a Tausand Tempico device and reads its settings.

    Instructions:
        * before running this example, pyTempico package must be installed.
        * change 'my_port' to your corresponding port.
        * run.

    | @author: David Guzman at Tausand Electronics
    | dguzman@tausand.com
    | https://www.tausand.com
"""

import pyTempico

my_port = 'COM17' #change this port to your Tempico device's port

my_device = pyTempico.TempicoDevice(my_port)    #create object
```

(continues on next page)

(continued from previous page)

```
print('opening connection with device in port',my_port)
my_device.open()             #open connection with device
if my_device.isOpen():
    print('connection with my_device is open')
else:
    print('connection with my_device is close')

print('\n1) reading general getSettings')
my_settings = my_device.getSettings()
print('settings:')
print(my_settings)

print('\n2) reading specific settings')
print('NumberOfRuns:', my_device.getNumberOfRuns())
print('ThresholdVoltage:', my_device.getThresholdVoltage())

print('\nreading channel specific settings')
print('Ch1 AverageCycles:', my_device.ch1.getAverageCycles())
print('Ch2 AverageCycles:', my_device.ch2.getAverageCycles())
print('Ch3 AverageCycles:', my_device.ch3.getAverageCycles())
print('Ch4 AverageCycles:', my_device.ch4.getAverageCycles())
print('Ch1 Mode:', my_device.ch1.getMode())
print('Ch1 NumberOfStops:', my_device.ch1.getNumberOfStops())
print('Ch1 isEnabled:', my_device.ch1.isEnabled())

print('\nclosing connection with device in port',my_port)
my_device.close()            #close connection with device
if my_device.isOpen():
    print('connection with my_device is open')
else:
    print('connection with my_device is close')
```

2.3 singleMeasureExample.py

```
# -*- coding: utf-8 -*-
"""singleMeasureExample

    Created on Tue May  7 09:23 2024

    Connects to a Tausand Tempico device, starts a measurement and reads the
    results.

    Instructions:
        * before running this example, pyTempico package must be installed.
        * change 'my_port' to your corresponding port.
        * connect signals to your Tempico Device. If no signals are measured,
        this example will return an empty data array.
        * run.
```

(continues on next page)

(continued from previous page)

```
    | @author: David Guzman at Tausand Electronics
    | dguzman@tausand.com
    | https://www.tausand.com
"""

import pyTempico

my_port = 'COM16' #change this port to your Tempico device's port

my_device = pyTempico.TempicoDevice(my_port)    #create object

print('opening connection with device in port',my_port)
my_device.open()             #open connection with device
if my_device.isOpen():
    print('connection with my_device is open')
else:
    print('connection with my_device is close')

print('sending a measure request to device')
my_device.measure()   #starts a measurement, and saves response in 'data'
data = my_device.measure()   #starts a measurement, and saves response in 'data'
print('measured data, in ps:',data)

my_device.close()            #close connection with device
if my_device.isOpen():
    print('connection with my_device is open')
else:
    print('connection with my_device is close')
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX