Skippylab Documentation

Release 0.7.0

Achim Stoessl

May 09, 2018

Contents

CHAPTER 1

Skippylab documentation contents

1.1 skippylab package

1.1.1 Subpackages

skippylab.instruments package

Submodules

skippylab.instruments.function_generators module

Function generators

class skippylab.instruments.function_generators.**Agilent33220AFunctionGenerator**(*ip='10.25.123.11*
gpib_address=15
loglevel=20)

Bases: object

skippylab.instruments.gpibcontrollers module

Wrapper around the PrologixGPIBEthernet adapter, to transparently manage instruments which are connected via GPIB and the respective controller

skippylab.instruments.gpibcontrollers.**prologix_gpib_ethernet_provider**(*ip*, *gpib_addres*)

Provide a vxi11 compatible instrument which is accesible transparently through its ip.

Parameters

- **ip** (*str*) – ip address of the controller
- **gpib_address** (*str*) – gpib adress of the instrument

Returns vxi11.instrument

skippylab.instruments.oscilloscopes module

Communicate with oscilloscope via vxi11 protocol over LAN network

class skippylab.instruments.oscilloscopes.**AbstractBaseOscilloscope**(*ip='169.254.68.19'*, *loglevel=20*)

Bases: object

A oscilloscope with a high sampling rate in the order of several gigasamples. Defines the scope API the DAQ reiles on

ACQUIRE_ONE = 'SEQ'

CONTINOUS_RUN = 'RUNSTop'

MAXTRIALS = 5

acquire_waveform()

ping()
Check if oscilloscope is connected

reopen_socket()
Close and reopen the socket after a timeout

Returns None

samplingrate
Get the current sampling rate

Returns float (GSamples/sec)

select_channel(*channel*)
Select a channel for the data acquisition

Parameters **channel** (*int*) – Channel number

Returns None

string_encoding = 'iso-8859-1'

class skippylab.instruments.oscilloscopes.**RhodeSchwarzRTO1044**(*ip*)
Bases: *skippylab.instruments.oscilloscopes.AbstractBaseOscilloscope*

Made by Rhode&Schwarz, scope with sampling rate up to 20GSamples/s

acquire_waveform()
Get the voltage values for a single waveform

Returns np.ndarray

do_single_acquisition()

run()
Start continuous acquisitions

Returns:

samplingrate
Get the current sampling rate

Returns float (GSamples/sec)

select_channel(*channel*)
Select the channel for the readout.

Parameters **channel** (*int*) – Channel number (1-4)

Returns None

stop()

triggerrate
Get the triggerrate of the scope

Parameters **interval** (*float*) – measurement time in seconds to

Returns float

class skippylab.instruments.oscilloscopes.**TektronixDPO4104B**(*ip*, *loglevel=20*)
Bases: *skippylab.instruments.oscilloscopes.AbstractBaseOscilloscope*

Oscilloscope of type DPO4104B manufactured by Tektronix

acquire

acquire_mode

acquire_waveform(*header=None*, *return_digitizer_levels=False*)
Get the waveform data

Keyword Arguments

- **header** (*dict*) – if header is None, a new header will be acquired
- **return_digitizer_levels** (*bool*) – return the waveform data in digitizer levels, not volts. Saves space for storage, since int8 can be used for 1-bit representation. ..and there is no float8 for obvious reasons.

Returns np.ndarray

average_waveform(*n=10*)
Acquire some waveforms and take the average

Keyword Args. n (int): number of waveforms to average over

Returns tuple(np.array). xs, ys

binary_formats = {'RI': '!b'}

binary_header_pattern = re.compile('#(?P<bin_head>[0-9]*)')

static convert_waveform(*header*, *waveform*)

data

data_start

data_stop

data_width

static decode_ascii_waveform(*response*)
Search the response for waveform data when in ascii format

Parameters **response** (*str*) – Hopefully the result of a CURVE command or similar

Returns np.ndarray

decode_binary_waveform(*response*, *header*)
Decaode a waveform in binary format. To do so, the header is required to know about the exact format.

Parameters

- **response** (*str*) – Hopefully the response to some CURVE command or similar

- **header** (*dict*) – A parsed waveform header

Returns np.ndarray

static decode_header(*response*, *return_last_index=False*, *absolute_timing=False*)
Parse a response searching for waveform header data

Parameters **head** (*str*) – hopefully the result of some WAVFrm or similar command

Keyword Arguments

- **return_last_index** (*bool*) – if True, also the last index of the header in the string will be returned
- **absolute_timing** (*bool*) – try to infer the absolute timeing (whatever that means) # FIXME!

Returns dict/tuple

fill_buffer()
Returns:

histbox

histend

histogram
Return a histogram which might be recorded by the scope

histstart

make_n_acquisitions(*n*, *trials=20*, *return_only_charge=False*, *single_acquisition=True*, *return_digitizer_levels=False*)
Acquire n waveforms

Parameters **n** (*int*) – Number of waveforms to acquire

Keyword Arguments

- **trials** (*int*) – Set breaking condition when to abort acquisition
- **return_only_charge** (*bool*) – don't get the wf, but only integrated charge instead
- **single_acquisition** (*bool*) – use the scopes single acquisition mode
- **return_digitizer_levels** (*bool*) – return the waveform data in digitizer levels, not volts. Saves space for storage, since int8 can be used for 1-bit representation. ..and there is no float8 for obvious reasons.

Returns [wf_1,wf_2,...]

Return type list

pull(*buff_header=True*, *use_buffered_acq_window=True*, *use_channel_info=True*)
Fit in the API for the DAQ. Returns waveform data

Keyword Arguments

- **buff_header** (*bool*) – buffer the header for subsequent acquisition without changing the parameters of the acquistion (much faster)
- **Default value of this should be False, however requires DAQ API change** (*FIXME!*) –
- **use_buffered_acq_window** (*bool*) – set this flag to cache the length of the acquisition window internally so that it does not get resetted when switching channels
- **use_channel_info** (*bool*) – select the channel on each submit

Returns dict

reset_acquisition_window()
Reset the acquisition window to the maximum possible acquisition window Returns:In

None

samplingrate
The samplingrate in GSamples/S

Returns float

select_channel(*channel*)
Select the channel for the readout

Parameters **channel** (*int*) – Channel number (1-4)

Returns None

set_acquisition_window(*start*, *stop*)
Set the acquisition window in bin number

Parameters

- **start** (*int*) – start bin
- **stop** (*int*) – stop bin

Returns None

set_acquisition_window_from_internal_buffer()
Use the internal buffer to set the data acquisition window. Might be necessary if the channel was switched in the meantime

Returns None

set_feature_acquisition_window(*leading*, *trailing*, *n_waveforms=20*)
Set the acquisition window around the most prominent feature in the waveform

Parameters

- **leading** (*float*) – leading ns before the most prominent feature
- **trailing** (*float*) – trailing ns after the most prominent feature

Keyword Args n_waveforms (int): average over n_waveforms to identify the most prominent feature

Returns None

set_waveform_encoding(*enc*)
Define the waveform encoding

Parameters **enc** –

Returns:

show_waveforms(*n=5*)
Demonstration function: Will use pylab show to plot some acquired waveforms

Keyword Arguments **n** (*int*) – number of waveforms to show

Returns None

source

time_binwidth
Get the binwidth of the time - that is sampling rate

Returns float

trigger_continuous()

trigger_frequency_enabled

trigger_single()

triggerrate
The rate the scope is triggering. The scope in principle provides this number, however we have to work around it as it does not work reliably

Keyword Arguments **interval** (*int*) – time interval to integrate measurement over in seconds

Returns float

waveform_bins
Get the time bin numbers for the waveform voltage data

Returns np.ndarray

waveform_enc

waveform_times
Get the time for the waveform bins

Returns np.ndarray

class skippylab.instruments.oscilloscopes.**UnknownOscilloscope**(*ip='169.254.68.19'*, *loglevel=20*)
Bases: *skippylab.instruments.oscilloscopes.AbstractBaseOscilloscope*

Use for testing and debugging

acquire_waveform()

samplingrate()
Get the current sampling rate

Returns float (GSamples/sec)

select_channel(*channel*)
Select a channel for the data acquisition

Parameters **channel** (*int*) – Channel number

Returns None

class skippylab.instruments.oscilloscopes.**Waveform**(*header*, *raw_waveform*)
Bases: object

A non-oscilloscope dependent representation of a measured waveform

load()

save()

time_bins()

volts()

skippylab.instruments.oscilloscopes.**get_header**(*self*)

skippylab.instruments.oscilloscopes.**set_header**(*self*, *header*)

skippylab.instruments.oscilloscopes.**setget**(*command*)
Shortcut to construct property object to wrap getters and setters for a number of settings

Parameters

- **command** (*str*) – The command being used to get/set. Get will be a query
- **value** (*str*) – The value to set

Returns property object

skippylab.instruments.powersupplies module

Connection to power supply unit

class skippylab.instruments.powersupplies.**KeysightE3631APowerSupply**(*ip='10.25.124.252'*, *gpib_address=5*, *loglevel=20*)
Bases: object

A low voltage power supply with two channels, +6V and +- 25V manufactured by Keysight. The power supply does not have an ethernet port, so the connection is done via GPIB and a prologix GPIB Ethernet connector

error_state
Read out the error register of the power supply

Returns str

measure_current(*channel*)
Measure current on givven channel

Parameters **channel** (*str*) –

Returns float

off()
Cut the power on all channels

Returns None

on()
Enable power on all channels

Returns None

output

ping()
Check the connection

Returns str

select_channel(*channel*)
Select either the +6, +25 or -25V channel

Parameters **channel** (*str or int*) –

Returns None

set_voltage(*channel*, *voltage*)
Set the supplied voltage of a channel to the desired value

Parameters

- **channel** (*str or int*) –

- **voltage** (*float*) –

Returns None

Module contents

skippylab.scpi package

Submodules

skippylab.scpi.commands module

A namespace for oscilloscope string commands. The commands are send as ASCII to the scope using a socket connection

class skippylab.scpi.commands.**KeysightE3631APowerSupplyCommands**
Bases: object

Namespace for the commands of the KeysightE3631APowerSupply

APPLY = 'APPLY'

CHANNEL = 'INST'

CURRENT = 'CURRENT'

DC = 'DC'

ERROR_STATEQ = 'SYST:ERR?'

MEASURE = 'MEASURE'

N25 = 'N25V'

OFF = 'OFF'

ON = 'ON'

OUTPUT = 'OUTPUT:STATE'

P25 = 'P25V'

P6 = 'P6V'

VOLT = 'VOLT'

class skippylab.scpi.commands.**RhodeSchwarzRTO1044Commands**
Bases: object

Namespace for the commands for the RhodeSchwarz oscilloscope

CH1 = 'CHAN1'

CH2 = 'CHAN2'

CH3 = 'CHAN3'

CH4 = 'CHAN4'

CURVE = 'DATA:VALues?'

N_ACQUISITONS = 'ACQuire:CURRent?'

RUN = 'RUN'

SINGLE = 'SINGLE'

STOP = 'STOP'

WAVEFORM = 'DATA?'

WF_HEADER = 'DATA:HEADer?'

class skippylab.scpi.commands.**TektronixDPO4104BCommands**
Bases: object

Namespace for the commands for the TektronixDP04104B

ACQUISITON_MODE = 'ACQuire:STOPAfter'

CH1 = 'CH1'

CH2 = 'CH2'

CH3 = 'CH3'

CH4 = 'CH4'

N_ACQUISITIONS = 'ACQuire:NUMACq?'

OFF = 'OFF'

ON = 'ON'

ONE = '1'

TRIGGER_FREQUENCYQ = 'TRIGger:FREQuency?'

TRIGGER_FREQUENCY_ENABLED = 'DISplay:TRIGFrequency'

WF = 'WAVFrm?'

WF_BYTEWIDTH = 'DATA:WIDTH'

WF_HEADER = 'WFMOutpre:BYT_Nr?;:WFMOutpre:ENCdg?; :WFMOutpre:NR_Pt?; :WFMOutpre:XZEro?

WF_NOHEAD = 'CURVE?'

ZERO = '0'

skippylab.scpi.commands.**add_arg**(*cmd*, *arg*)
Add an argument to a command string. Concat the two.

Parameters

- **cmd** (*str*) – The base command
- **arg** (*str*) – An argument

Returns str

skippylab.scpi.commands.**clean_response**(*response*)
Remove some EOL remainders from the resulting scope response

Parameters **response** (*str*) – response from the scope

Returns str

skippylab.scpi.commands.**concat**(**cmd*)
Combine several commands

Parameters **cmd** – list of commands

Returns str

skippylab.scpi.commands.**histbox_coordinates**(*left*, *top*, *right*, *bottom*)
Create a string for the box coordinates for the histogram set up by the scope itself. The result can be send to the scope to set the box.

Parameters

- **left** (*int*) –
- **top** (*int*) –
- **right** (*int*) –
- **bottom** (*int*) –

Returns str

skippylab.scpi.commands.**parse_curve_data**(*header*, *curve*)
Make sense out of that what is returned by CURVE. This works only if the scope is set to return the data in ASCII, not binary.

Parameters

- **header** (*dict*) – a parsed header
- **curv** (*str*) – returned by CURVE?

Returns np.ndarray xs, np.ndarray values

Return type tuple

skippylab.scpi.commands.**query**(*cmd*)

Module contents

1.1.2 Submodules

1.1.3 skippylab.daq module

Use the scope as a DAQ

class skippylab.daq.**DAQ**
Bases: object

A virtual DAQ using an oscilloscope

acquire(**pullargs*, ***pullkwargs*)
Go through the instrument list and trigger their pull methods to build an event

Keyword Arguments ****pullkwargs** (*dict*) – will be passed on the individual pull methods

Returns pyosci.Event

acquire_n_events(*n_events*, *trigger_hook=<function DAQ.<lambda>>*, *trigger_hook_args=(None,)*, *pull_args=()*, *pull_kwargs={}*)
Continuous acquisition. Acquires n events. Yields events. Use trigger hook to define a function to decide when data is returned.

Parameters

- **n_events** (*int*) – Number of events to acquire
- **trigger_hook** (*callable*) – Trigger condition
- **trigger_hook_args** (*tuple*) – Arguments for the trigger condition

Yields Event

register_instrument(*instrument*, *label='instrument'*)
Register an instrument and assign a channel to it. Instruments must have a pull() method which allows to pull data from them at a certain event.

Parameters

- **instrument** (*ducktype*) – needs to be configured already and must have a pull() method
- **channel_name** (*int*) – identify the instrument under this registered channel

Returns None

class skippylab.daq.**Event**(*use_datetime=False*)
Bases: object

DAQ will return events when triggered.

timestamp_it()
Give it a timestamp! Time in seconds

Returns None

1.1.4 skippylab.loggers module

Prepare logging functionality for the module

skippylab.loggers.**get_logger**(*loglevel*, *logfile=None*)
A root logger with a formatted output logging to stdout and a file

Parameters

- **loglevel** (*int*) – 10,20,30,... the higher the less logging
- **logfile** (*str*) – write logging to this file as well as stdout

Returns logging.logger

1.1.5 skippylab.plotting module

Convenient plot functions

skippylab.plotting.**plot_histogram**(*bincenters*, *bincontent*, *fig=None*, *savename='test.png'*, *remove_empty_bins=True*)
Plot a histogram returned by TektronixDPO4104B.get_histogram Use pylab.plot

Parameters

- **bincenters** (*np.ndarray); bincenters (x*) –
- **bincontent** (*np.ndarray*) – bincontent (y)

Keyword Arguments

- **fig** (*pylab.figure*) – A figure instance
- **savename** (*str*) – where to save the figure (full path)
- **remove_empty_bins** (*bool*) – Cut away preceeding and trailing zero bins

skippylab.plotting.**plot_waveform**(*wf_header*, *wf_data*, *fig=None*, *savename=None*, *use_mv_and_ns=True*, *color=None*)

Make a plot of a single acquisition

Parameters

- **wf_header** (*dict*) – custom waveform header
- **wf_data** (*np.ndarray*) – waveform data

Keyword Arguments

- **fig** (*pylab.figure*) – A figure instance
- **savename** (*str*) – where to save the figure (full path)
- **use_mv_and_ns** (*bool*) – use mV and ns instead of V and s

Returns pylab.fig

1.1.6 skippylab.tools module

Convenient operations

skippylab.tools.**average_wf**(*waveforms*)

Get the average waveform

Parameters **waveforms** (*iterable of np.ndarrays*) –

Returns np.ndarray

skippylab.tools.**integrate_wf**(*waveform*, *xs*, *xstep*, *method=<Mock name='mock.integrate.simps' id='140338442310376'>*, *impedance=50*)

Integrate a waveform, i.e. a voltage curve. If the desired result shall be indeed a charge, please make sure to give xs in seconds and impedance in Ohm accordingly. xstep needs to be in seconds as well.

Parameters

- **waveform** (*np.ndarray*) – voltage values
- **xs** (*np.ndarray*) – timing values
- **xstep** (*float*) – timing bin size

Keyword Arguments

- **method** (*func*) – integration method
- **impedance** (*float*) – needed to calculate actual charge

Returns float

skippylab.tools.**load_waveform**(*filename*, *converter=<function <lambda>>*)

load a waveform from a file

Parameters **filenaame** (*str*) – An existing filename

Keyword Arguments **converter** (*func*) – If the data is saved in digitizer levels, use the converter function to convert to Volts

Returns tuple (dict, np.ndarray)

skippylab.tools.**save_waveform**(*header*, *waveform*, *filename*)

save a waveform together with its header

Parameters

- **header** (*dict*) – Some metainformation about the waveform
- **waveform** (*np.ndarray*) – the actual voltage data
- **filename** (*str*) – a filename where the data should be saved

Returns None

1.1.7 Module contents

Package to read out TektronixDPO4104B oscilloscope

CHAPTER 2

Indices and tables

- modindex
- Glossary

Python Module Index

S

Index

T

U

V

W

Z