# **PyMeasure Documentation**

Release 0.15.1.dev366+gcb643a9c2.d20250827

**PyMeasure Developers** 

# **LEARNING PYMEASURE**

1	Introduction1.1 Instrument ready1.2 Graphical displays	3 3
2	Quick start2.1Setting up Python2.2Installing PyMeasure	<b>5</b> 5 5
3	Tutorials  3.1 Connecting to an instrument	7 9 18
4	4.1 Adapter base class	<b>47</b> 47 49 51 53
5	5.1       Experiment class       6         5.2       Listener class       6         5.3       Procedure class       6         5.4       Parameter classes       6         5.5       Worker class       7	63 64 65 67 71
6	6.1       Browser classes         6.2       Console class         6.3       Curves classes         6.4       Inputs classes         6.5       Listeners classes         6.6       Log classes         6.7       Manager classes         6.8       Plotter class         6.9       Qt classes         6.10       Thread classes         6.11       Widget classes	<b>75</b> 75 76 77 79 79 81 81 81

7	pyme	easure.instruments	97
	7.1	Instrument classes	97
	7.2	Generic Instrument Types Mixins	108
	7.3	Validator functions	108
	7.4	Comedi data acquisition	110
	7.5	Resource Manager	111
	7.6	Active Technologies	111
	7.7	Aculight	116
	7.8	Advantest	117
	7.9	Agilent	149
	7.10	Aim-TTI	215
	7.11	AJA International	218
	7.12	Ametek	220
	7.13	AMI	222
	7.14	Anaheim Automation	224
	7.15	Anapico	226
	7.16	Andeen Hagerling	227
	7.17	Anritsu	230
	7.18	Attocube	245
	7.19	BK Precision	248
	7.20	Danfysik	249
	7.21	Delta Elektronika	252
	7.22	Edwards	253
	7.23	EURO TEST	253
	7.24	Fluke	256
		F.W. Bell	
		Heidenhain	
	7.27	HC Photonics	
		Hewlett Packard	
		Inficon	
		IPG Photonics	
	7.31	Keithley	
		KEPCO INC.	
		Keysight	
		Kuhne Electronic	
		Lake Shore Cryogenics	
		LeCroy	
		MKS Instruments	
		Newport	476
	7.39		478
	7.40	Novanta Photonics	492
	7.41	Oxford Instruments	493 505
	7.42 7.43	Parker	506
	7.43	Philips	507
	7.44	Proterial	509
	7.45	PTW Freiburg	511
	7.47	Racal-Dana	519
	7.48		521
	7.48	Razorbill	521
	7.49	Rigol	524
	7.51	Rohde & Schwarz	526
	7.52	Santec	544
	7.53	Siglent Technologies	_
			2.5

	7.54	Signal Recovery	551
	7.55	Spellman HV	564
	7.56	Stanford Research Systems	568
	7.57	T&C Power Conversion	585
	7.58	TDK Lambda	589
	7.59	Tektronix	598
	7.60	Teledyne	
	7.61	Temptronic	
	7.62	TEXIO	
	7.63	Thermotron	
	7.64	Thorlabs	
	7.65	Thyracont	
	7.66	Toptica	
	7.67	Velleman	
	7.68	Yokogawa	
	7.08	Tokogawa	037
8	Cont	ributing	649
U	8.1	Using the development version	
	8.2	Working on a new feature	
		· · · · · · · · · · · · · · · · · · ·	
	8.3	Making a pull request	
	8.4	Unit testing	651
9	Dono	orting an error	653
,	керо	itting an error	033
10	Addi	ng instruments	655
		File structure	
	10.2	Instrument file	
	10.3	Your instrument's user interface	
	10.4	Defining default connection settings	
	10.5	Writing properties	
	10.5	Instruments with similar features	
	10.0	Instruments with channels	
	10.8	Advanced communication protocols	
		Writing tests	
	10.10	Solutions for implementation challenges	680
11	Codi	ng Standards	681
11		Python style guides	
		Documentation	
	11.3	Usage of getter and setter functions	
	11.4	Docstrings	682
12	Auth	ors	683
-	1 Lucii		000
13	Licen	nse	687
14		ngelog	689
		8	689
	14.2	-,	689
	14.3	, , , , , , , , , , , , , , , , , , , ,	691
	14.4	Version 0.13.1 (2023-10-05)	694
	14.5	Version 0.13.0 (2023-09-23)	
	14.6	Version 0.12.0 (2023-07-05)	695
	14.7	Version 0.11.1 (2022-12-31)	699
	14.8	Version 0.11.0 (2022-11-19)	699

14.9 Version 0.10.0 (2022-04-09)	701
14.10 Version 0.9 – released 2/7/21	704
14.11 Version 0.8 – released 3/29/19	705
14.12 Version 0.7 – released 8/4/19	705
14.13 Version 0.6.1 – released 4/21/19	706
14.14 Version 0.6 – released 1/14/19	706
14.15 Version 0.5.1 – released 4/14/18	706
14.16 Version 0.5 – released 10/18/17	706
14.17 Version 0.4.6 – released 8/12/17	707
14.18 Version 0.4.5 – released 7/4/17	707
14.19 Version 0.4.4 – released 6/4/17	707
14.20 Version 0.4.3 – released 3/30/17	707
14.21 Version 0.4.2 – released 8/23/16	708
14.22 Version 0.4.1 – released 7/31/16	
14.23 Version 0.4 – released 7/29/16	708
14.24 Version 0.3 – released 4/8/16	708
14.25 Version 0.2 – released 12/16/15	709
14.26 Version 0.1.6 – released 4/19/15	
14.27 Version 0.1.5 – release 10/22/14	709
14.28 Version 0.1.4 – released 8/2/14	709
14.29 Version 0.1.3 – released 7/20/14	709
14.30 Version 0.1.2 – released 7/18/14	
14.31 Version 0.1.1 – released 7/16/14	
14.32 Version 0.1.0 – released 7/15/14	
15 Related Projects	711
Python Module Index	713
Index	715



PyMeasure makes scientific measurements easy to set up and run. The package contains a repository of instrument classes and a system for running experiment procedures, which provides graphical interfaces for graphing live data and managing queues of experiments. Both parts of the package are independent, and when combined provide all the necessary requirements for advanced measurements with only limited coding.

Installing Python and PyMeasure are demonstrated in the *Quick Start guide*. From there, checkout the existing *instruments that are available for use*.

PyMeasure is currently under active development, so please report any issues you experience on our Issues page.

The main documentation for the site is organized into a couple sections:

- Learning PyMeasure
- API Reference
- About PyMeasure

Information about development is also available:

· Getting involved

**LEARNING PYMEASURE** 

**CHAPTER** 

ONE

## INTRODUCTION

PyMeasure uses an object-oriented approach for communicating with scientific instruments, which provides an intuitive interface where the low-level SCPI and GPIB commands are hidden from normal use. Users can focus on solving the measurement problems at hand, instead of re-inventing how to communicate with instruments.

Instruments with VISA (GPIB, Serial, etc) are supported through the PyVISA package under the hood. Prologix GPIB adapters are also supported. Communication protocols can be swapped, so that instrument classes can be used with all supported protocols interchangeably.

In order to keep the corresponding numbers and physical units (e.g. 5 meters) together, pint quantities can be used. That way it is easy to handle different orders of magnitude (meters and centimeters) or different units (meters and feet).

Before using PyMeasure, you may find it helpful to be acquainted with basic Python programming for the sciences and understand the concept of objects.

## 1.1 Instrument ready

The package includes a number of *instruments already defined*. Their definitions are organized based on the manufacturer name of the instrument. For example the class that defines the *Keithley 2400 SourceMeter* can be imported by calling:

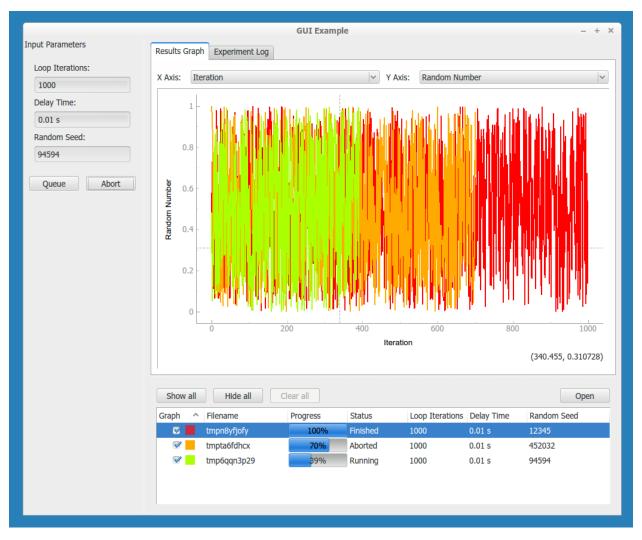
from pymeasure.instruments.keithley import Keithley2400

The *Tutorials* section will go into more detail on *connecting to an instrument*. If you don't find the instrument you are looking for, but are interested in contributing, see the documentation on *adding an instrument*.

# 1.2 Graphical displays

Graphical user interfaces (GUIs) can be easily generated to manage execution of measurement procedures with PyMeasure. This includes live plotting for data, and a queue system for managing large numbers of experiments.

These features are explored in the *Using a graphical interface* tutorial.



The GUIs are not restricted to the instruments included in this package. Any python instrument may be used. For example, this script demonstrates how to use an InstrumentKit instrument.

**CHAPTER** 

**TWO** 

## **QUICK START**

This section provides instructions for getting up and running quickly with PyMeasure.

# 2.1 Setting up Python

The easiest way to install the necessary Python environment for PyMeasure is through the Anaconda distribution, which includes 720 scientific packages. The advantage of using this approach over just relying on the pip installer is that Anaconda correctly installs the required Qt libraries.

Download and install the appropriate Python version of Anaconda for your operating system.

# 2.2 Installing PyMeasure

#### 2.2.1 Install with conda

If you have the Anaconda distribution you can use the conda package manager to easily install PyMeasure and all required dependencies.

Open a terminal and type the following commands (on Windows look for the *Anaconda Prompt* in the Start Menu):

```
conda config --add channels conda-forge
conda install pymeasure
```

This will install PyMeasure and all the required dependencies.

## 2.2.2 Install with pip

PyMeasure can also be installed with pip.

```
pip install pymeasure
```

Depending on your operating system, using this method may require additional work to install the required dependencies, which include the Qt libraries.

### 2.2.3 Installing VISA

Typically, communication with your instrument will happen using PyVISA, which is installed automatically. However, this needs a VISA implementation installed to handle device communication. If you do not already know what this means, install the pure-Python pyvisa-py package (using the same installation you used above). If you want to know more, consult the PyVISA documentation.

## 2.2.4 Checking the version

Now that you have Python and PyMeasure installed, open your python environment (e.g. a REPL or Jupyter notebook) to test which version you have installed. Execute the following Python code.

```
import pymeasure
pymeasure.__version__
```

You should see the version of PyMeasure printed out. At this point you have PyMeasure installed, and you are ready to start using it! Are you ready to *connect to an instrument*?

**CHAPTER** 

THREE

## **TUTORIALS**

The following sections provide instructions for getting started with PyMeasure.

## 3.1 Connecting to an instrument

After following the *Quick Start* section, you now have a working installation of PyMeasure. This section describes connecting to an instrument, using a Keithley 2400 SourceMeter as an example. To follow the tutorial, open a command prompt, IPython terminal, or Jupyter notebook.

First import the instrument of interest.

```
from pymeasure.instruments.keithley import Keithley2400
```

Then construct an object by passing the VISA address. For this example we connect to the instrument over GPIB (using VISA) with an address of 4:

```
sourcemeter = Keithley2400("GPIB::4")
```

#### 1 Note

Passing an appropriate resource string is the default method when creating pymeasure instruments. See the *adapters* section below for more details.

If you are not sure about the correct resource string identifying your instrument, you can run the pymeasure. instruments.resources.list\_resources() function to list all available resources:

```
from pymeasure.instruments.resources import list_resources
list_resources()
```

If you know the USB properties (vendor id, product id, serial numer) of the serial device, you can query for the VISA resource string:

For instruments with standard SCPI commands, an id property will return the results of a \*IDN? SCPI command, identifying the instrument.

```
sourcemeter.id
```

This is equivalent to manually calling the SCPI command.

```
sourcemeter.ask("*IDN?")
```

Here the ask method writes the SCPI command, reads the result, and returns that result. This is further equivalent to calling the methods below.

```
sourcemeter.write("*IDN?")
sourcemeter.read()
```

This example illustrates that the top-level methods like id are really composed of many lower-level methods. Both can be called depending on the operation that is desired. PyMeasure hides the complexity of these lower-level operations, so you can focus on the bigger picture.

Instruments are also equipped to be used in a with statement.

```
with Keithley2400("GPIB::4") as sourcemeter:
    sourcemeter.id
```

When the with-block is exited, the shutdown method of the instrument will be called, turning the system into a safe state.

```
with Keithley2400("GPIB::4") as sourcemeter:
    sourcemeter.isShutdown == False
sourcemeter.isShutdown == True
```

## 3.1.1 Using adapters

PyMeasure supports a number of adapters, which are responsible for communicating with the underlying hardware. In the example above, we passed the string "GPIB::4" when constructing the instrument. By default this constructs a VISAAdapter (our most popular, default adapter) to connect to the instrument using VISA. Passing a string (or integer in case of GPIB) is by far the most typical way to create pymeasure instruments.

Sometimes, you might need to go beyond the usual setup, which is also possible. Instead of passing a string, you could equally pass an adapter object.

```
from pymeasure.adapters import VISAAdapter

adapter = VISAAdapter("GPIB::4")
sourcemeter = Keithely2400(adapter)
```

To instead use a Prologix GPIB device connected on /dev/ttyUSB0 (proper permissions are needed in Linux, see *PrologixAdapter*), the adapter is constructed in a similar way. The Prologix adapter can be shared by many instruments. Therefore, new *PrologixAdapter* instances with different GPIB addresses can be generated from an already existing instance.

```
from pymeasure.adapters import PrologixAdapter

adapter = PrologixAdapter('ASRL/dev/ttyUSB0::INSTR', address=7)
sourcemeter = Keithley2400(adapter) # at GPIB address 7
multimeter = Keithley2000(adapter.gpib(9)) # at GPIB address 9
```

## 3.1.2 Modifying connection settings

Sometimes you want to tweak the connection settings when talking to a device. This might be because you have a non-standard device or connection, or are troubleshooting why a device does not reply.

When using a string or integer to connect to an instrument, a *VISAAdapter* is used internally. Additional settings need to be passed in as keyword arguments. For example, to use a fast baud rate on a quick connection when connecting to the Keithely2400 as above, do

```
sourcemeter = Keithley2400("ASRL2", timeout=500, baud_rate=115200)
```

This overrides any defaults that may be defined for the instrument, either generally valid ones like timeout or interface-specific ones like baud\_rate.

If you use an invalid argument, either misspelled or not valid for the chosen interface, an exception will be raised.

When using a separately-created Adapter instance, you define any custom settings when creating the adapter. Any keyword arguments passed in are discarded.

The above examples illustrate different methods for communicating with instruments, using adapters to keep instrument code independent from the communication protocols. Next we present the methods for setting up measurements.

# 3.2 Making a measurement

This tutorial will walk you through using PyMeasure to acquire a current-voltage (IV) characteristic using a Keithley 2400. Even if you don't have access to this instrument, this tutorial will explain the method for making measurements with PyMeasure. First we describe using a simple script to make the measurement. From there, we show how *Procedure* objects greatly simplify the workflow, which leads to making the measurement with a graphical interface.

## 3.2.1 Using scripts

Scripts are a quick way to get up and running with a measurement in PyMeasure. For our IV characteristic measurement, we perform the following steps:

- 1) Import the necessary packages
- 2) Set the input parameters to define the measurement
- 3) Set source\_current and measure\_voltage parameters
- 4) Connect to the Keithley 2400
- 5) Set up the instrument for the IV characteristic
- 6) Allocate arrays to store the resulting measurements
- 7) Loop through the current points, measure the voltage, and record
- 8) Save the final data to a CSV file
- 9) Shutdown the instrument

These steps are expressed in code as follows.

```
# Import necessary packages
from pymeasure.instruments.keithley import Keithley2400
import numpy as np
import pandas as pd
from time import sleep
# Set the input parameters
data_points = 50
averages = 10
max\_current = 0.001
min_current = -max_current
# Set source_current and measure_voltage parameters
current_range = 10e-3 # in Amps
compliance_voltage = 10 # in Volts
measure_nplc = 0.1 # Number of power line cycles
voltage_range = 1 # in VOlts
# Connect and configure the instrument
sourcemeter = Keithley2400("GPIB::24")
sourcemeter.reset()
sourcemeter.use_front_terminals()
sourcemeter.apply_current(current_range, compliance_voltage)
sourcemeter.measure_voltage(measure_nplc, voltage_range)
sleep(0.1) # wait here to give the instrument time to react
sourcemeter.stop_buffer()
sourcemeter.disable_buffer()
# Allocate arrays to store the measurement results
currents = np.linspace(min_current, max_current, num=data_points)
voltages = np.zeros_like(currents)
voltage_stds = np.zeros_like(currents)
sourcemeter.enable_source()
# Loop through each current point, measure and record the voltage
for i in range(data_points):
    sourcemeter.config_buffer(averages)
    sourcemeter.source_current = currents[i]
   sourcemeter.start_buffer()
   sourcemeter.wait_for_buffer()
    # Record the average and standard deviation
   voltages[i] = sourcemeter.means[0]
    sleep(1.0)
   voltage_stds[i] = sourcemeter.standard_devs[0]
# Save the data columns in a CSV file
data = pd.DataFrame({
    'Current (A)': currents,
    'Voltage (V)': voltages,
    'Voltage Std (V)': voltage_stds,
})
data.to_csv('example.csv')
                                                                            (continues on next page)
```

```
sourcemeter.shutdown()
```

Running this example script will execute the measurement and save the data to a CSV file. While this may be sufficient for very basic measurements, this example illustrates a number of issues that PyMeasure solves. The issues with the script example include:

- The progress of the measurement is not transparent
- Input parameters are not associated with the data that is saved
- Data is not plotted during the execution (nor at all in this case)
- Data is only saved upon successful completion, which is otherwise lost
- Canceling a running measurement causes the system to end in a undetermined state
- Exceptions also end the system in an undetermined state

The *Procedure* class allows us to solve all of these issues. The next section introduces the *Procedure* class and shows how to modify our script example to take advantage of these features.

## 3.2.2 Using Procedures

The Procedure object bundles the sequence of steps in an experiment with the parameters required for its successful execution. This simple structure comes with huge benefits, since a number of convenient tools for making the measurement use this common interface.

Let's start with a simple example of a procedure which loops over a certain number of iterations. We make the SimpleProcedure object as a sub-class of Procedure, since SimpleProcedure *is a* Procedure.

```
from time import sleep
from pymeasure.experiment import Procedure
from pymeasure.experiment import IntegerParameter
class SimpleProcedure(Procedure):
    # a Parameter that defines the number of loop iterations
   iterations = IntegerParameter('Loop Iterations')
    # a list defining the order and appearance of columns in our data file
   DATA_COLUMNS = ['Iteration']
    def execute(self):
        """Execute the procedure.
        Loops over each iteration and emits the current iteration.
        before waiting for 0.01 sec, and then checking if the procedure
        should stop.
        11 11 11
        for i in range(self.iterations):
            self.emit('results', {'Iteration': i})
            sleep(0.01)
            if self.should_stop():
                break
```

At the top of the SimpleProcedure class we define the required Parameters. In this case, iterations is a IntegerParameter that defines the number of loops to perform. Inside our Procedure class we reference the value in the iterations Parameter by the class variable where the Parameter is stored (self.iterations). PyMeasure swaps out the Parameters with their values behind the scene, which makes accessing the values of parameters very convenient.

We define the data columns that will be recorded in a list stored in DATA\_COLUMNS. This sets the order by which columns are stored in the file. In this example, we will store the Iteration number for each loop iteration.

The execute methods defines the main body of the procedure. Our example method consists of a loop over the number of iterations, in which we emit the data to be recorded (the Iteration number). The data is broadcast to any number of listeners by using the emit method, which takes a topic as the first argument. Data with the 'results' topic and the proper data columns will be recorded to a file. The sleep function in our example provides two very useful features. The first is to delay the execution of the next lines of code by the time argument in units of seconds. The seconds is that during this delay time, the CPU is free to perform other code. Successful measurements often require the intelligent use of sleep to deal with instrument delays and ensure that the CPU is not hogged by a single script. After our delay, we check to see if the Procedure should stop by calling self.should\_stop(). By checking this flag, the Procedure will react to a user canceling the procedure execution.

## **1** Note

Instead of emitting results one by one, it is also possible to emit results in batch. As an example consider a device that returns multiple points for each measurement (such as an oscilloscope or a CCD):

```
intensities = self.ccd.capture() # Assume this function returns a list of_
intensities for each pixel
pixels = np.arange(len(intensities))

for pixel, intensity in zip(pixels, intensities):
    self.emit('results', {'Pixel': pixel, 'Intensity': intensity})
```

The downside to this method is that it is cumbersome to write and can be slow when the array of data is large. Instead it is possible to emit the data as as a whole:

```
intensities = self.ccd.capture() # Assume this function returns a list of
    intensities for each pixel
pixels = np.arange(len(intensities))
self.emit('batch results', {'Pixel': pixels, 'Intensity': intensities})
Please note that you have to use 'batch results' as the topic when emitting results this way.
```

This covers the basic requirements of a Procedure object. Now let's construct our SimpleProcedure object with 100 iterations.

```
procedure = SimpleProcedure()
procedure.iterations = 100
```

Next we will show how to run the procedure.

#### **Running Procedures**

A Procedure is run by a Worker object. The Worker executes the Procedure in a separate Python thread, which allows other code to execute in parallel to the procedure (e.g. a graphical user interface). In addition to performing the measurement, the Worker spawns a Recorder object, which listens for the 'results' topic in data emitted by the Procedure, and writes those lines to a data file. The Results object provides a convenient abstraction to keep track of where the data should be stored, the data in an accessible form, and the Procedure that pertains to those results.

We first construct a Results object for our Procedure.

```
from pymeasure.experiment import Results

data_filename = 'example.csv'
results = Results(procedure, data_filename)
```

Constructing the Results object for our Procedure creates the file using the data\_filename, and stores the Parameters for the Procedure. This allows the Procedure and Results objects to be reconstructed later simply by loading the file using Results.load(data\_filename). The Parameters in the file are easily readable.

We now construct a Worker with the Results object, since it contains our Procedure.

```
from pymeasure.experiment import Worker
worker = Worker(results)
```

The Worker publishes data and other run-time information through specific queues, but can also publish this information over the local network on a specific TCP port (using the optional port argument. Using TCP communication allows great flexibility for sharing information with Listener objects. Queues are used as the standard communication method because they preserve the data order, which is of critical importance to storing data accurately and reacting to the measurement status in order.

Now we are ready to start the worker.

```
worker.start()
```

This method starts the worker in a separate Python thread, which allows us to perform other tasks while it is running. When writing a script that should block (wait for the Worker to finish), we need to join the Worker back into the main thread.

```
worker.join(timeout=3600) # wait at most 1 hr (3600 sec)
```

Let's put all the pieces together. Our SimpleProcedure can be run in a script by the following.

```
from time import sleep
from pymeasure.experiment import Procedure, Results, Worker
from pymeasure.experiment import IntegerParameter

class SimpleProcedure(Procedure):

    # a Parameter that defines the number of loop iterations
    iterations = IntegerParameter('Loop Iterations')

# a list defining the order and appearance of columns in our data file
    DATA_COLUMNS = ['Iteration']

def execute(self):
```

```
"""Execute the procedure.
        Loops over each iteration and emits the current iteration,
        before waiting for 0.01 sec, and then checking if the procedure
        should stop.
        for i in range(self.iterations):
            self.emit('results', {'Iteration': i})
            sleep(0.01)
            if self.should_stop():
                break
if __name__ == "__main__":
   procedure = SimpleProcedure()
   procedure.iterations = 100
   data_filename = 'example.csv'
   results = Results(procedure, data_filename)
   worker = Worker(results)
   worker.start()
   worker.join(timeout=3600) # wait at most 1 hr (3600 sec)
```

Here we have included an if statement to only run the script if the \_\_name\_\_ is \_\_main\_\_. This precaution allows us to import the SimpleProcedure object without running the execution.

#### **Using Logs**

Logs keep track of important details in the execution of a procedure. We describe the use of the Python logging module with PyMeasure, which makes it easy to document the execution of a procedure and provides useful insight when diagnosing issues or bugs.

Let's extend our SimpleProcedure with logging.

```
import logging
log = logging.getLogger(__name__)
log.addHandler(logging.NullHandler())

from time import sleep
from pymeasure.log import console_log
from pymeasure.experiment import Procedure, Results, Worker
from pymeasure.experiment import IntegerParameter

class SimpleProcedure(Procedure):
    iterations = IntegerParameter('Loop Iterations')

DATA_COLUMNS = ['Iteration']

def execute(self):
    log.info("Starting the loop of %d iterations" % self.iterations)
```

```
for i in range(self.iterations):
            data = {'Iteration': i}
            self.emit('results', data)
            log.debug("Emitting results: %s" % data)
            sleep(0.01)
            if self.should_stop():
                log.warning("Caught the stop flag in the procedure")
                break
if __name__ == "__main__":
   console_log(log)
   log.info("Constructing a SimpleProcedure")
   procedure = SimpleProcedure()
   procedure.iterations = 100
   data_filename = 'example.csv'
   log.info("Constructing the Results with a data file: %s" % data_filename)
   results = Results(procedure, data_filename)
   log.info("Constructing the Worker")
   worker = Worker(results)
   worker.start()
   log.info("Started the Worker")
   log.info("Joining with the worker in at most 1 hr")
   worker.join(timeout=3600) # wait at most 1 hr (3600 sec)
   log.info("Finished the measurement")
```

First, we have imported the Python logging module and grabbed the logger using the \_\_name\_\_ argument. This gives us logging information specific to the current file. Conversely, we could use the '' argument to get all logs, including those of pymeasure. We use the console\_log function to conveniently output the log to the console. Further details on how to use the logger are addressed in the Python logging documentation.

#### Storing metadata

Metadata (pymeasure.experiment.parameters.Metadata) allows storing information (e.g. the actual starting time, instrument parameters) about the measurement in the header of the datafile. These Metadata objects are evaluated and stored in the datafile only after the startup method has run; this way it is possible to e.g. retrieve settings from an instrument and store them in the file. Using a Metadata is nearly as straightforward as using a Parameter; extending the example of above to include metadata, looks as follows:

```
from time import sleep, time
from pymeasure.experiment import Procedure
from pymeasure.experiment import IntegerParameter, Metadata

class SimpleProcedure(Procedure):

# a Parameter that defines the number of loop iterations
iterations = IntegerParameter('Loop Iterations')
```

```
# the Metadata objects store information after the startup has ran
starttime = Metadata('Start time', fget=time)
custom_metadata = Metadata('Custom', default=1)
# a list defining the order and appearance of columns in our data file
DATA_COLUMNS = ['Iteration']
def startup(self):
    self.custom_metadata = 20
def execute(self):
    """ Loops over each iteration and emits the current iteration,
    before waiting for 0.01 sec, and then checking if the procedure
    1111111
    for i in range(self.iterations):
        self.emit('results', {'Iteration': i})
        sleep(0.01)
        if self.should_stop():
            break
```

As with a Parameter, PyMeasure swaps out the Metadata with their values behind the scene, which makes accessing the values of Metadata very convenient.

The value of a Metadata can be set either using an fget method or manually in the startup method. The fget method, if provided, is run after startup method. It can also be provided as a string; in that case it is assumed that the string contains the name of an attribute (either a callable or not) of the Procedure class which returns the value that is to be stored. This also allows to retrieve nested attributes (e.g. in order to store a property or method of an instrument) by separating the attributes with a period: e.g. <code>instrument\_name.attribute\_name</code> (or even <code>instrument\_name.subclass\_name.attribute\_name</code>); note that here only the final element (i.e. <code>attribute\_name</code> in the example) is allowed to refer to a callable. If neither an fget method is provided or a value manually set, the Metadata will return to its default value, if set. The formatting of the value of the Metadata-object can be controlled using the <code>fint</code> argument.

#### **Modifying our script**

16

Now that you have a background on how to use the different features of the Procedure class, and how they are run, we will revisit our IV characteristic measurement using Procedures. Below we present the modified version of our example script, now as a IVProcedure class.

```
# Import necessary packages
from pymeasure.instruments.keithley import Keithley2400
from pymeasure.experiment import Procedure, Results, Worker
from pymeasure.experiment import IntegerParameter, FloatParameter
from time import sleep
import numpy as np

from pymeasure.log import log, console_log

class IVProcedure(Procedure):
    data_points = IntegerParameter('Data points', default=20)
    averages = IntegerParameter('Averages', default=8)
```

```
max_current = FloatParameter('Maximum Current', units='A', default=0.001)
   min_current = FloatParameter('Minimum Current', units='A', default=-0.001)
   DATA_COLUMNS = ['Current (A)', 'Voltage (V)', 'Voltage Std (V)']
   def startup(self):
        log.info("Connecting and configuring the instrument")
        self.sourcemeter = Keithley2400("GPIB::24")
        self.sourcemeter.reset()
        self.sourcemeter.use_front_terminals()
        self.sourcemeter.apply_current(100e-3, 10.0) # current_range = 100e-3,
self.sourcemeter.measure_voltage(0.01, 1.0) # nplc = 0.01, voltage\_range = 1.0
        sleep(0.1) # wait here to give the instrument time to react
        self.sourcemeter.stop_buffer()
        self.sourcemeter.disable_buffer()
   def execute(self):
        currents = np.linspace(
            self.min_current,
            self.max_current,
            num=self.data points
        self.sourcemeter.enable_source()
        # Loop through each current point, measure and record the voltage
        for current in currents:
            self.sourcemeter.config_buffer(IVProcedure.averages.value)
            log.info("Setting the current to %g A" % current)
            self.sourcemeter.source_current = current
            self.sourcemeter.start_buffer()
            log.info("Waiting for the buffer to fill with measurements")
            self.sourcemeter.wait_for_buffer()
            data = {
                'Current (A)': current,
                'Voltage (V)': self.sourcemeter.means[0],
                'Voltage Std (V)': self.sourcemeter.standard_devs[0]
            self.emit('results', data)
            sleep(0.01)
            if self.should_stop():
                log.info("User aborted the procedure")
                break
    def shutdown(self):
        self.sourcemeter.shutdown()
        log.info("Finished measuring")
if __name__ == "__main__":
   console_log(log)
    log.info("Constructing an IVProcedure")
```

```
procedure = IVProcedure()
procedure.data_points = 20
procedure.averages = 8
procedure.max_current = -0.001
procedure.min_current = 0.001

data_filename = 'example.csv'
log.info("Constructing the Results with a data file: %s" % data_filename)
results = Results(procedure, data_filename)

log.info("Constructing the Worker")
worker = Worker(results)
worker.start()
log.info("Started the Worker")

log.info("Joining with the worker in at most 1 hr")
worker.join(timeout=3600) # wait at most 1 hr (3600 sec)
log.info("Finished the measurement")
```

The parentheses in the COLUMN entries indicate the physical unit of the data in the corresponding column, e.g. 'Voltage Std (V)' indicates Volts. If you want to indicate a dimensionless value, e.g. Mach number, you can use (1) instead. Combined units like (m/s) or the long form (meter/second) are also possible. The class Results ensures, that the data is stored in the correct unit, here Volts. For example a pint. Quantity of 500 mV will be stored as 0.5 V. A string will be converted first to a Quantity and a mere number (e.g. float, int, ...) is assumed to be already in the right unit (e.g. 5 will be stored as 5 V). If the data entry is not compatible, either because it has the wrong unit, e.g. meters which is not a unit of voltage, or because it is no number at all, a warning is logged and 'nan' will be stored in the file. If you do not specify a unit (i.e. no parentheses), no unit check is performed for this column, unless the data entry is a Quantity for that column. In this case, this column's unit is set to the base unit (e.g. meter if unit of the data entry is kilometers) of the data entry. From this point on, unit checks are enabled for this column. Also use columns without unit checks (i.e. without parentheses) for strings or booleans.

At this point, you are familiar with how to construct a Procedure sub-class. The next section shows how to put these procedures to work in a graphical environment, where will have live-plotting of the data and the ability to easily queue up a number of experiments in sequence. All of these features come from using the Procedure object.

# 3.3 Using a graphical interface

In the previous tutorial we measured the IV characteristic of a sample to show how we can set up a simple experiment in PyMeasure. The real power of PyMeasure comes when we also use the graphical tools that are included to turn our simple example into a full-fledged user interface.

## 3.3.1 Using the Plotter

While it lacks the nice features of the ManagedWindow, the Plotter object is the simplest way of getting live-plotting. The Plotter takes a Results object and plots the data at a regular interval, grabbing the latest data each time from the file.

#### **A** Warning

The example in this section is known to raise issues when executed: a *QApplication was not created in the main thread / nextEventMatchingMask should only be called from the Main Thread* warning is raised. While the example works without issues on some operating systems and python configurations, users are advised not to rely on the plotter while this issue is unresolved. Users can hence skip this example and continue with the *Using the ManagedWindow* section.

Let's extend our SimpleProcedure with a RandomProcedure, which generates random numbers during our loop. This example does not include instruments to provide a simpler example.

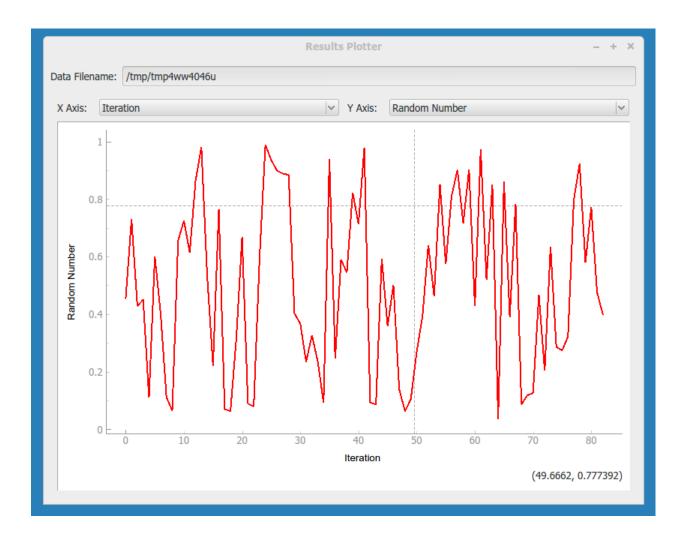
```
import logging
log = logging.getLogger(__name__)
log.addHandler(logging.NullHandler())
import random
from time import sleep
from pymeasure.log import console_log
from pymeasure.display import Plotter
from pymeasure.experiment import Procedure, Results, Worker
from pymeasure.experiment import IntegerParameter, FloatParameter, Parameter
class RandomProcedure(Procedure):
   iterations = IntegerParameter('Loop Iterations')
   delay = FloatParameter('Delay Time', units='s', default=0.2)
    seed = Parameter('Random Seed', default='12345')
   DATA_COLUMNS = ['Iteration', 'Random Number']
   def startup(self):
        log.info("Setting the seed of the random number generator")
        random.seed(self.seed)
    def execute(self):
        log.info("Starting the loop of %d iterations" % self.iterations)
        for i in range(self.iterations):
            data = {
                'Iteration': i,
                'Random Number': random.random()
            self.emit('results', data)
            log.debug("Emitting results: %s" % data)
            self.emit('progress', 100 * i / self.iterations)
            sleep(self.delay)
            if self.should_stop():
```

```
log.warning("Caught the stop flag in the procedure")
                break
if name == " main ":
   console_log(log)
   log.info("Constructing a RandomProcedure")
   procedure = RandomProcedure()
   procedure iterations = 100
   data_filename = 'random.csv'
   log.info("Constructing the Results with a data file: %s" % data_filename)
   results = Results(procedure, data_filename)
   log.info("Constructing the Plotter")
   plotter = Plotter(results)
   plotter.start()
   log.info("Started the Plotter")
   log.info("Constructing the Worker")
   worker = Worker(results)
   worker.start()
   log.info("Started the Worker")
   log.info("Joining with the worker in at most 1 hr")
   worker.join(timeout=3600) # wait at most 1 hr (3600 sec)
   log.info("Finished the measurement")
```

The important addition is the construction of the Plotter from the Results object.

```
plotter = Plotter(results)
plotter.start()
```

The Plotter is started in a different process so that it can be run on a separate CPU for higher performance. The Plotter launches a Qt graphical interface using pyqtgraph which allows the Results data to be viewed based on the columns in the data.



## 3.3.2 Using the ManagedWindow

The ManagedWindow is the most convenient tool for running measurements with your Procedure. This has the major advantage of accepting the input parameters graphically. From the parameters, a graphical form is automatically generated that allows the inputs to be typed in. With this feature, measurements can be started dynamically, instead of defined in a script.

Another major feature of the ManagedWindow is its support for running measurements in a sequential queue. This allows you to set up a number of measurements with different input parameters, and watch them unfold on the liveplot. This is especially useful for long running measurements. The ManagedWindow achieves this through the Manager object, which coordinates which Procedure the Worker should run and keeps track of its status as the Worker progresses.

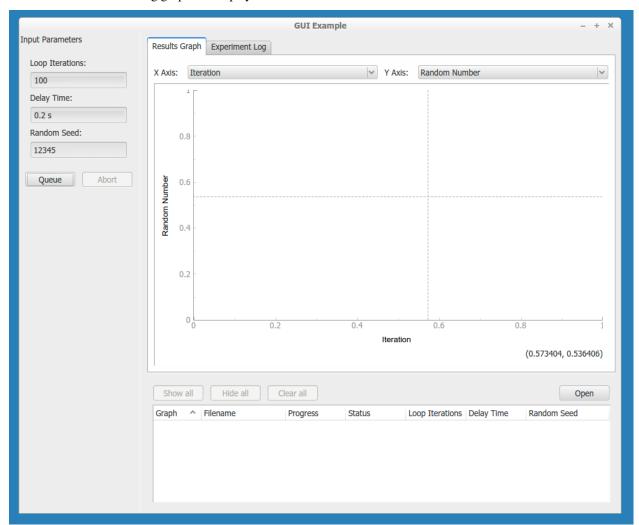
Below we adapt our previous example to use a ManagedWindow.

```
import logging
log = logging.getLogger(__name__)
log.addHandler(logging.NullHandler())

import sys
import tempfile
import random
from time import sleep

(continues on next page)
```

```
from pymeasure.log import console_log
from pymeasure.display.Qt import QtWidgets
from pymeasure.display.windows import ManagedWindow
from pymeasure.experiment import Procedure, Results
from pymeasure.experiment import IntegerParameter, FloatParameter, Parameter
class RandomProcedure(Procedure):
   iterations = IntegerParameter('Loop Iterations', default=100)
    delay = FloatParameter('Delay Time', units='s', default=0.2)
    seed = Parameter('Random Seed', default='12345')
   DATA_COLUMNS = ['Iteration', 'Random Number']
   def startup(self):
        log.info("Setting the seed of the random number generator")
        random.seed(self.seed)
   def execute(self):
        log.info("Starting the loop of %d iterations" % self.iterations)
        for i in range(self.iterations):
            data = {
                'Iteration': i,
                'Random Number': random.random()
            }
            self.emit('results', data)
            log.debug("Emitting results: %s" % data)
            self.emit('progress', 100 * i / self.iterations)
            sleep(self.delay)
            if self.should_stop():
                log.warning("Caught the stop flag in the procedure")
                break
class MainWindow(ManagedWindow):
    def __init__(self):
        super().__init__(
            procedure_class=RandomProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            x_axis='Iteration',
            y_axis='Random Number'
        self.setWindowTitle('GUI Example')
if __name__ == "__main__":
   app = QtWidgets.QApplication(sys.argv)
   window = MainWindow()
   window.show()
    sys.exit(app.exec())
```



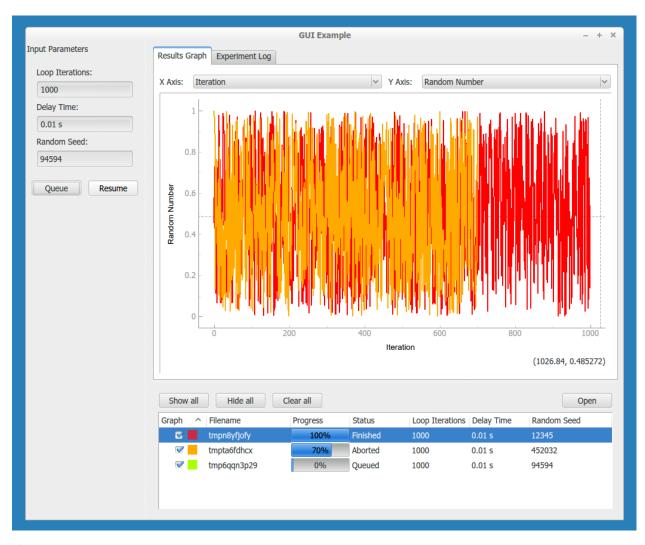
This results in the following graphical display.

In the code, the MainWindow class is a sub-class of the ManagedWindow class. We override the constructor to provide information about the procedure class and its options. The inputs are a list of Parameters class-variable names, which the display will generate graphical fields for. When the list of inputs is long, a boolean key-word argument inputs\_in\_scrollarea is provided that adds a scrollbar to the input area. The displays is a list similar to the inputs list, which instead defines the parameters to display in the browser window. This browser keeps track of the experiments being run in the sequential queue.

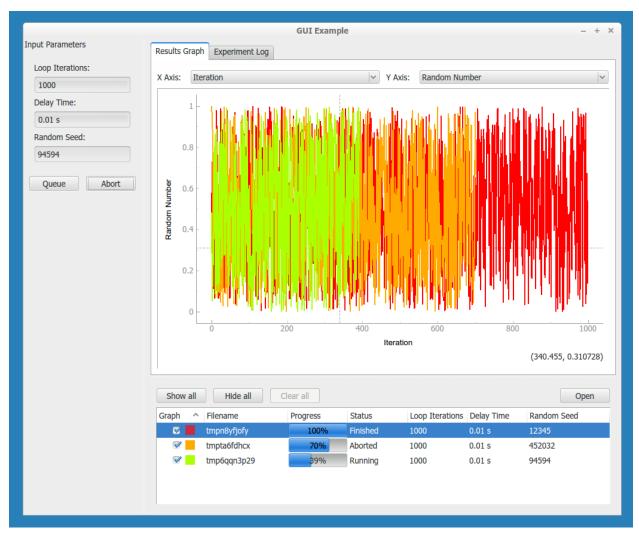
As a bit of background information (for basic usage this needs not be known): the *queue()* method establishes how the *Procedure* object is constructed. The make\_procedure() method is used to create a *Procedure* based on the graphical input fields. Here we are free to modify the procedure before putting it on the queue. In this context, the *Manager* uses an *Experiment* object to keep track of the *Procedure*, *Results*, and its associated graphical representations in the browser and live-graph. This is then given to the *Manager* to queue the experiment.



By default the Manager starts a measurement when its procedure is queued. The abort button can be pressed to stop an experiment. In the Procedure, the self.should\_stop call will catch the abort event and halt the measurement. It is important to check this value, or the Procedure will not be responsive to the abort event.



If you abort a measurement, the resume button must be pressed to continue the next measurement. This allows you to adjust anything, which is presumably why the abort was needed.



Now that you have learned about the ManagedWindow, you have all of the basics to get up and running quickly with a measurement and produce an easy to use graphical interface with PyMeasure.

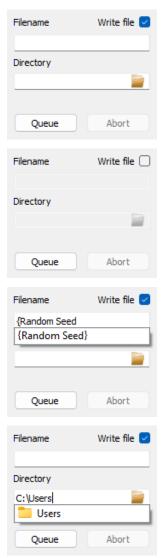


For performance reasons, the default linewidth of all the graphs has been set to 1. If performance is not an issue, the linewidth can be changed to 2 (or any other value) for better visibility by using the *linewidth* keyword-argument in the *Plotter* or the *ManagedWindow*. Whenever a linewidth of 2 is preferred and a better performance is required, it is possible to enable using OpenGL in the import section of the file:

```
import pyqtgraph as pg
pg.setConfigOption("useOpenGL", True)
```

#### The filename and directory input

By default, a ManagedWindow instance contains fields for the filename and the directory (as part of the FileInputWidget) to control where the results of an experiment are saved.



If the checkbox named *Save data* is enabled, the measurement is written to a file. Otherwise, it is stored in a temporary file.

The filename in the designated field can be entered with or without extension. If the entered extension is recognized (by default .csv and .txt are recognized), that extension is used. If the extension is not recognized, the first of the available extensions will be used (default is .csv). Additionally, a sequence number is added just before the extension to ensure the uniqueness of the filename.

The filename can also contain placeholders, which are filled in using the standard python format function (i.e., placeholders can be entered as '{placeholder name:formatspec}'). Valid placeholders are the names of the all the input parameters or metadata of the measurement procedure; the valid placeholders are listed in the tooltip of the input field. As the standard format functionality is used, the placeholders can be formatted as such; for example, the filename 'DATA\_delay{Delay Time:08.3f}s' gets formatted into 'DATA\_delay0000.010s'.

Both the filename and the directory field are provided with auto-completion to help with filling in these fields. The directory field contains a button on the right side to open a folder-selection window.

The default values can be easily set after the *ManagedWindow* has been initialized; this allows setting a default location and a default filename, changing the default recognized extensions, control the default toggle-value for the *Save data* option, and control whether the filename input field is frozen.

```
class MainWindow(ManagedWindow):
    def __init__(self):
        super().__init__(
            procedure_class=TestProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            x_axis='Iteration',
            y_axis='Random Number',
        )
        self.setWindowTitle('GUI Example')
        self.filename = r'default_filename_delay{Delay Time:4f}s'
                                                                      # Sets default.
→ filename
        self.directory = r'C:/Path/to/default/directory'
                                                                      # Sets default.
\rightarrowdirectory
        self.store_measurement = False
                                                                      # Controls the 'Save.
→data' toggle
        self.file_input.extensions = ["csv", "txt", "data"]
                                                                      # Sets recognized_
→extensions, first entry is the default extension
        self.file_input.filename_fixed = False
                                                                       # Controls whether.
→ the filename-field is frozen (but still displayed)
```

The presence of the widget is controlled by the boolean argument enabled\_file\_input of the <code>ManagedWindow</code> init. Note that when this is set to False, the default <code>queue()</code> method of the <code>ManagedWindow</code> class will no longer work, and a new, custom, method needs to be implemented; a basic implementation is shown in the documentation of the <code>queue()</code> method.

## 3.3.3 Customising the plot options

For both the PlotterWindow and ManagedWindow, plotting is provided by the pyqtgraph library. This library allows you to change various plot options, as you might expect: axis ranges (by default auto-ranging), logarithmic and semilogarithmic axes, downsampling, grid display, FFT display, etc. There are two main ways you can do this:

- 1. You can right click on the plot to manually change any available options. This is also a good way of getting an overview of what options are available in pyqtgraph. Option changes will, of course, not persist across a restart of your program.
- 2. You can programmatically set these options using pyqtgraph's PlotItem API, so that the window will open with these display options already set, as further explained below.

For *Plotter*, you can make a sub-class that overrides the <code>setup\_plot()</code> method. This method will be called when the Plotter constructs the window. As an example

```
class LogPlotter(Plotter):
    def setup_plot(self, plot):
        # use logarithmic x-axis (e.g. for frequency sweeps)
        plot.setLogMode(x=True)
```

For ManagedWindow, the mechanism to customize plots is much more flexible by using specialization via inheritance. Indeed ManagedWindowBase is the base class for ManagedWindow and ManagedImageWindow which are subclasses

ready to use for GUI.

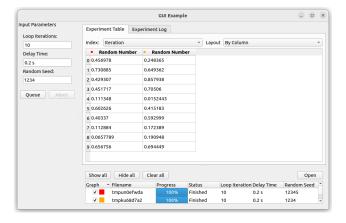
## 3.3.4 Using tabular format

In some experiments, data in tabular format may be useful in addition or in alternative to graphical plot. ManagedWindowBase allows adding a TableWidget to show experiments data, the widget supports also exporting data in some popular format like CSV, HTML, etc. Below an example on how to customize ManagedWindowBase to use tabular format, it derived from example above and changed lines are marked.

```
import logging
log = logging.getLogger(__name__)
log.addHandler(logging.NullHandler())
import sys
import tempfile
import random
from time import sleep
from pymeasure.log import console log
from pymeasure.display.Qt import QtWidgets
from pymeasure.display.windows import ManagedWindowBase
from pymeasure.display.widgets import TableWidget, LogWidget
from pymeasure.experiment import Procedure, Results
from pymeasure.experiment import IntegerParameter, FloatParameter, Parameter
class RandomProcedure(Procedure):
    iterations = IntegerParameter('Loop Iterations', default=10)
    delay = FloatParameter('Delay Time', units='s', default=0.2)
    seed = Parameter('Random Seed', default='12345')
    DATA_COLUMNS = ['Iteration', 'Random Number']
    def startup(self):
        log.info("Setting the seed of the random number generator")
        random.seed(self.seed)
    def execute(self):
        log.info("Starting the loop of %d iterations" % self.iterations)
        for i in range(self.iterations):
            data = {
                'Iteration': i.
                'Random Number': random.random()
            }
            self.emit('results', data)
            log.debug("Emitting results: %s" % data)
            self.emit('progress', 100 * i / self.iterations)
            sleep(self.delay)
            if self.should_stop():
                log.warning("Caught the stop flag in the procedure")
                break
class MainWindow(ManagedWindowBase):
                                                                           (continues on next page)
```

```
def __init__(self):
        widget_list = (TableWidget("Experiment Table",
                                    RandomProcedure DATA_COLUMNS,
                                    by_column=True,
                                    ),
                       LogWidget("Experiment Log"),
        super().__init__(
            procedure_class=RandomProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            widget_list=widget_list,
        logging.getLogger().addHandler(widget_list[1].handler)
        log.setLevel(self.log_level)
        log.info("ManagedWindow connected to logging")
        self.setWindowTitle('GUI Example')
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec())
```

This results in the following graphical display.



## 3.3.5 Defining your own ManagedWindow's widgets

The parameter widget\_list in <code>ManagedWindowBase</code> constructor allow to introduce user's defined widget in the GUI results display area. The user's widget should inherit from <code>TabWidget</code> and could reimplement any of the methods that needs customization. In order to get familiar with the mechanism, users can check the following widgets already provided:

- LogWidget
- PlotWidget
- ImageWidget

- DockWidget
- TableWidget

## 3.3.6 Using the sequencer

As an extension to the way of graphically inputting parameters and executing multiple measurements using the <code>ManagedWindow</code>, <code>SequencerWidget</code> is provided which allows users to queue a series of measurements with varying one, or more, of the parameters. This sequencer thereby provides a convenient way to scan through the parameter space of the measurement procedure.

To activate the sequencer, two additional keyword arguments are added to <code>ManagedWindow</code>, namely sequencer and sequencer\_inputs. sequencer accepts a boolean stating whether or not the sequencer has to be included into the window and sequencer\_inputs accepts either <code>None</code> or a list of the parameter names are to be scanned over. If no list of parameters is given, the parameters displayed in the manager queue are used.

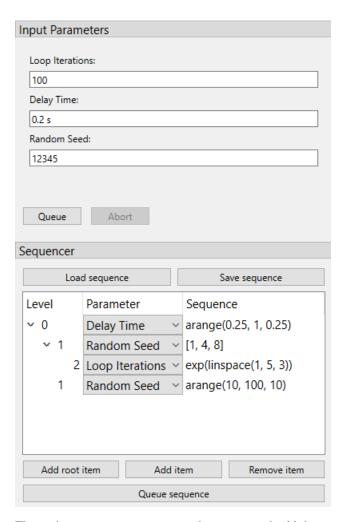
In order to be able to use the sequencer, the <code>ManagedWindow</code> class is required to have a queue method which takes a keyword (or better keyword-only for safety reasons) argument <code>procedure</code>, where a procedure instance can be passed. The sequencer will use this method to queue the parameter scan.

In order to implement the sequencer into the previous example, only the *ManagedWindow* has to be modified slightly (where modified lines are marked):

```
class MainWindow(ManagedWindow):

    def __init__(self):
        super().__init__(
            procedure_class=TestProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            x_axis='Iteration',
            y_axis='Random Number',
            sequencer=True,
            sequencer_inputs=['iterations', 'delay', 'seed'], # Added line
            sequence_file="gui_sequencer_example_sequence.txt", # Added line, optional
    )
    self.setWindowTitle('GUI Example')
```

This adds the sequencer underneath the input panel.



The widget contains a tree-view where you can build the sequence. It has three columns: level (indicated how deep an item is nested), parameter (a drop-down menu to select which parameter is being sequenced by that item), and sequence (the text-box where you can define the sequence). While the two former columns are rather straightforward, filling in the later requires some explanation.

In order to maintain flexibility, the sequence is defined in a text-box, allowing the user to enter any list-generating single-line piece of code. To assist in this, a number of functions is supported, either from the main python library (namely range, sorted, and list) or the numpy library. The supported numpy functions (prepending numpy. or any abbreviation is not required) are: arange, linspace, arccos, arcsin, arctan, arctan2, ceil, cos, cosh, degrees, e, exp, fabs, floor, fmod, frexp, hypot, ldexp, log, log10, modf, pi, power, radians, sin, sinh, sqrt, tan, and tanh.

As an example, arange(0, 10, 1) generates a list increasing with steps of 1, while using exp(arange(0, 10, 1)) generates an exponentially increasing list. This way complex sequences can be entered easily.

The sequences can be extended and shortened using the buttons Add root item, Add item, and Remove item. The latter two either add an item as a child of the currently selected item or remove the selected item, respectively. To queue the entered sequence the button Queue sequence can be used. If an error occurs in evaluating the sequence text-boxes, this is mentioned in the logger, and nothing is queued.

Finally, it is possible to create a sequence file such that the user does not need to write the sequence again each time. The sequence file can be created by saving current sequence built within the GUI using the Save sequence button or directly writing a simple text file. Once created, the sequence can be loaded with the Load sequence button.

In the sequence file each line adds one item to the sequence tree, starting with a number of dashes (-) to indicate the

32 Chapter 3. Tutorials

level of the item (starting with 1 dash for top level), followed by the name of the parameter and the sequence string, both as a python string between parentheses.

An example of such a sequence file is given below, resulting in the sequence shown in the figure above.

```
- "Delay Time", "arange(0.25, 1, 0.25)"
-- "Random Seed", "[1, 4, 8]"
--- "Loop Iterations", "exp(linspace(1, 5, 3))"
-- "Random Seed", "arange(10, 100, 10)"
```

This file can also be automatically loaded at the start of the program by adding the key-word argument sequence\_file="filename.txt" to the super().\_\_init\_\_ call, as was done in the example.

## 3.3.7 Using the estimator widget

In order to provide estimates of the measurement procedure, an *EstimatorWidget* is provided that allows the user to define and calculate estimates. The widget is automatically activated when the <code>get\_estimates</code> method is added in the <code>Procedure</code>.

The quickest and most simple implementation of the get\_estimates function simply returns the estimated duration of the measurement in seconds (as an int or a float). As an example, in the example provided in the *Using the ManagedWindow* section, the Procedure is changed to:

```
class RandomProcedure(Procedure):
    # ...

def get_estimates(self, sequence_length=None, sequence=None):
    return self.iterations * self.delay
```

This will add the estimator widget at the dock on the left. The duration and finishing-time of a single measurement is always displayed in this case. Depending on whether the SequencerWidget is also used, the length, duration and finishing-time of the full sequence is also shown.

For maximum flexibility (e.g. for showing multiple and other types of estimates, such as the duration, filesize, finishing-time, etc.) it is also possible that the <code>get\_estimates</code> returns a list of tuples. Each of these tuple consists of two strings: the first is the name (label) of the estimate, the second is the estimate itself.

As an example, in the example provided in the *Using the ManagedWindow* section, the Procedure is changed to:

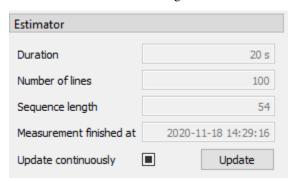
(continues on next page)

(continued from previous page)

]

return estimates

This will add the estimator widget at the dock on the left.



Note that after the initialisation of the widget both the label of the estimate as of course the estimate itself can be modified, but the amount of estimates is fixed.

The keyword arguments are not required in the implementation of the function, but are passed if asked for (i.e. def get\_estimates(self) does also works). Keyword arguments that are accepted are sequence, which contains the full sequence of the sequencer (if present), and sequence\_length, which gives the length of the sequence as integer (if present). If the sequencer is not present or the sequence cannot be parsed, both sequence and sequence\_length will contain None.

The estimates are automatically updated every 2 seconds. Changing this update interval is possible using the "Update continuously"-checkbox, which can be toggled between three states: off (i.e. no updating), auto-update every two seconds (default) or auto-update every 100 milliseconds. Manually updating the estimates (useful whenever continuous updating is turned off) is also possible using the "update"-button.

# 3.3.8 Flexible hiding of inputs

There can be situations when it may be relevant to turn on or off a number of inputs (e.g. when a part of the measurement script is skipped upon turning of a single BooleanParameter). For these cases, it is possible to assign a Parameter to a controlling Parameter, which will hide or show the Input of the Parameter depending on the value of the Parameter. This is done with the group\_by key-word argument.

```
toggle = BooleanParameter("toggle", default=True)
param = FloatParameter('some parameter', group_by='toggle')
```

When both the toggle and param are visible in the InputsWidget (via inputs=['iterations', 'delay', 'seed'] as demonstrated above) one can control whether the input-field of param is visible by checking and unchecking the checkbox of toggle. By default, the group will be visible if the value of the group\_by Parameter is True (which is only relevant for a BooleanParameter), but it is possible to specify other value as conditions using the group\_condition keyword argument.

```
iterations = IntegerParameter('Loop Iterations', default=100)
param = FloatParameter('some parameter', group_by='iterations', group_condition=99)
```

Here the input of param is only visible if iterations has a value of 99. This works with any type of Parameter as group\_by parameter.

To allow for even more flexibility, it is also possible to pass a (lambda)function as a condition:

Now the input of param is only shown if the value of iterations is between 51 and 99.

Using the hide\_groups keyword-argument of the ManagedWindow you can choose between hiding the groups (hide\_groups = True) and disabling / graying-out the groups (hide\_groups = False).

Finally, it is also possible to provide multiple parameters to the group\_by argument, in which case the input will only be visible if all of the conditions are true. Multiple parameters for grouping can either be passed as a dict of string: condition pairs, or as a list of strings, in which case the group\_condition can be either a single condition or a list of conditions:

Note that in this example, param\_A and param\_B are identically grouped: they're only visible if iterations is between 51 and 99 and if the *toggle* checkbox is checked (i.e. True).

# 3.3.9 Using the ManagedDockWindow

Building off the *Using the ManagedWindow* section where we used a ManagedWindow, we can also use *ManagedDockWindow* to build a graphical interface with multiple graphs that can be docked in the main GUI window or popped out into their own window.

To start with, let's make the following highlighted edits to the code example from *Using the ManagedWindow*:

- 1. On line 10 we now import ManagedDockWindow
- 2. On line 20, and lines 32 and 33, we add two new columns of data to be recorded 'Random Number 2' and 'Random Number 3'
- 3. On line 44 we make MainWindow a subclass of ManagedDockWindow
- 4. On line 51 we will pass in a list of strings from DATA\_COLUMNS to the x\_axis argument
- 5. On line 52 we will pass in a list of strings from DATA\_COLUMNS to the y\_axis argument

```
import logging
log = logging.getLogger(__name__)
log.addHandler(logging.NullHandler())

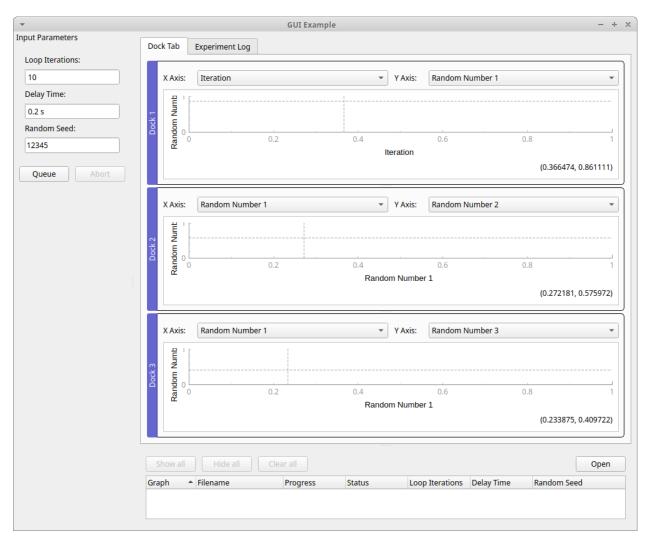
import sys
import tempfile
import random
from time import sleep
from pymeasure.display.Qt import QtWidgets
from pymeasure.display.windows.managed_dock_window import ManagedDockWindow
from pymeasure.experiment import Procedure, Results
from pymeasure.experiment import IntegerParameter, FloatParameter, Parameter
```

(continues on next page)

(continued from previous page)

```
class RandomProcedure(Procedure):
   iterations = IntegerParameter('Loop Iterations', default=10)
   delay = FloatParameter('Delay Time', units='s', default=0.2)
    seed = Parameter('Random Seed', default='12345')
   DATA_COLUMNS = ['Iteration', 'Random Number 1', 'Random Number 2', 'Random Number 3']
   def startup(self):
        log.info("Setting the seed of the random number generator")
        random.seed(self.seed)
   def execute(self):
        log.info("Starting the loop of %d iterations" % self.iterations)
        for i in range(self.iterations):
            data = {
                'Iteration': i,
                'Random Number 1': random.random(),
                'Random Number 2': random.random(),
                'Random Number 3': random.random()
            self.emit('results', data)
            log.debug("Emitting results: %s" % data)
            self.emit('progress', 100 * i / self.iterations)
            sleep(self.delay)
            if self.should_stop():
                log.warning("Caught the stop flag in the procedure")
                break
class MainWindow(ManagedDockWindow):
   def __init__(self):
        super().__init__(
            procedure_class=RandomProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            x_axis=['Iteration', 'Random Number 1'],
            y_axis=['Random Number 1','Random Number 2', 'Random Number 3']
        self.setWindowTitle('GUI Example')
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
   window = MainWindow()
   window.show()
    sys.exit(app.exec())
```

Now we can see our ManagedDockWindow:



As you can see from the above screenshot, our example code created three docks with following "X Axis" and "Y Axis" labels:

- 1. X Axis: "Iteration" Y Axis: "Random Number 1"
- 2. X Axis: "Random Number 1" Y Axis: "Random Number 2"
- 3. X Axis: "Random Number 1" Y Axis: "Random Number 3"

The list of strings for  $x_axis$  and  $y_axis$  set the default labels for each dockable plot and the longest list determines how many dockable plots are created. To highlight this point, in our example we define  $x_axis$  and  $y_axis$  with the following lists:

```
x_axis=['Iteration', 'Random Number 1'],
y_axis=['Random Number 1','Random Number 2', 'Random Number 3']
```

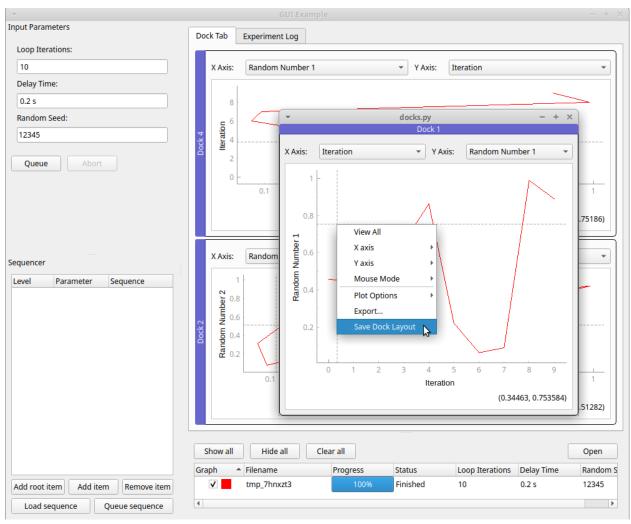
If one list is longer than the last element if the other list is used as the default label for the rest of the dockable plots. In our example that is why we have two **X Axis** labels with "Random Number 1". The longest list between **x\_axis** and **y\_axis** determines the number of plots. In our example **y\_axis** has the longest list with a length of three so three plots are created.

You can pop out a dockable plot from the main dock window to its own window by double clicking the blue "Dock #" title bar, which is to the left of each plot by default:

You can return the popped out window to the main window by clicking the close icon X in the top right.

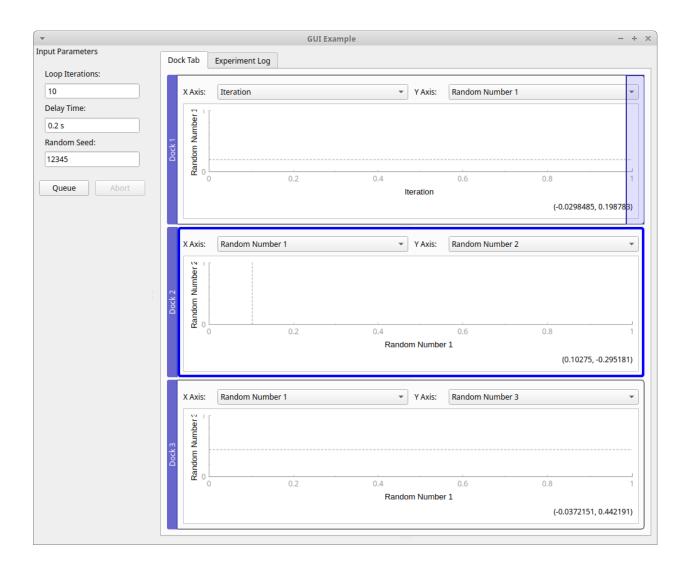
After positioning your dock windows, you can save the layout by right-clicking a dock widget and select "Save Dock Layout" from the context menu. This will save the layout of all docks and the settings for each plot to a file. By default the file path is the current working directory of the python file that started ManagedDockWindow, and the default file name is 'procedure class + "\_dock\_layout.json"". For our example, that would be "./RandomProcedure\_dock\_layout.json"

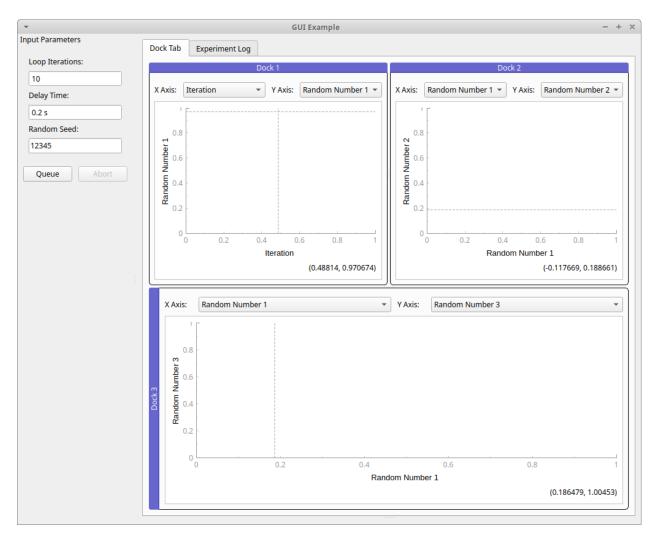
When you run the python file that invokes ManagedDockWindow again, it will look for and load the dock layout file if it exists.



You can drag a dockable plot to reposition it in reference to other plots in the main dock window in several ways. You can drag the blue "Dock #" title bar to the left or right side of another plot to reposition a plot to be side by side with another plot:

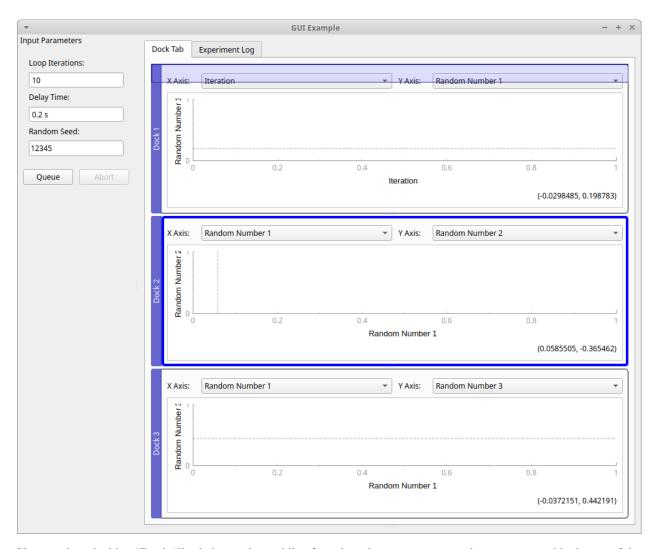
38 Chapter 3. Tutorials



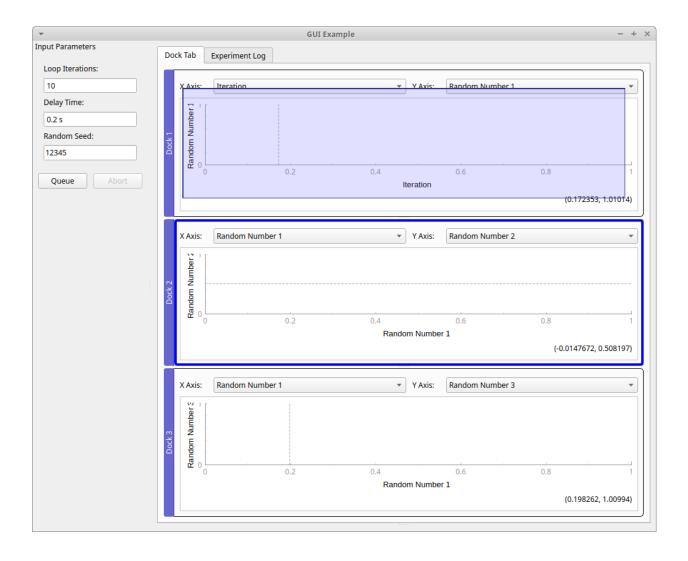


You can also drag the blue "Dock #" title bar to the top or bottom side of another plot to reposition a plot to rearrange the vertical order of the plots:

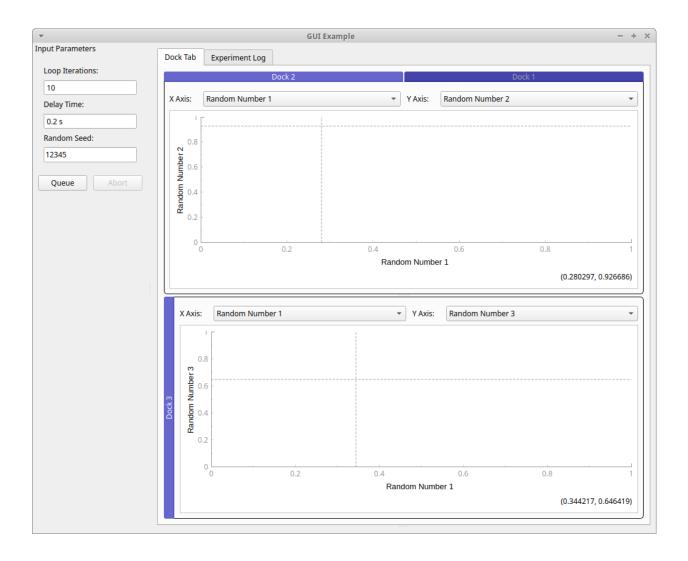
40 Chapter 3. Tutorials



You can drag the blue "Dock #" title bar to the middle of another plot to reposition a plot to create a tabbed view of the two plots:



42 Chapter 3. Tutorials



# 3.3.10 Using the ManagedConsole

The <code>ManagedConsole</code> is the most convenient tool for running measurements with your Procedure using a command line interface. The <code>ManagedConsole</code> allows to run an experiment with the same set of parameters available in the <code>ManagedWindow</code>, but they are defined using a set of command line switches.

It is also possible to define a test that uses both <code>ManagedConsole</code> or <code>ManagedWindow</code> according to user selection in the command line.

Enabling console mode is easy and straightforward and the following example demonstrates how to do it.

The following example is a variant of the code example from *Using the ManagedWindow* where some parts have been highlighted:

- 1. On line 8 we now import ManagedConsole
- 2. On line 73, we add the support for console mode

```
import sys
import random
import tempfile
from time import sleep
```

(continued from previous page)

```
from pymeasure.experiment import Procedure, IntegerParameter, Parameter, FloatParameter
from pymeasure.experiment import Results
from pymeasure.display.console import ManagedConsole
from pymeasure.display.Qt import QtWidgets
from pymeasure.display.windows import ManagedWindow
import logging
log = logging.getLogger('')
log.addHandler(logging.NullHandler())
class TestProcedure(Procedure):
    iterations = IntegerParameter('Loop Iterations', default=100)
   delay = FloatParameter('Delay Time', units='s', default=0.2)
    seed = Parameter('Random Seed', default='12345')
   DATA_COLUMNS = ['Iteration', 'Random Number']
   def startup(self):
        log.info("Setting up random number generator")
        random.seed(self.seed)
   def execute(self):
        log.info("Starting to generate numbers")
        for i in range(self.iterations):
            data = {
                'Iteration': i,
                'Random Number': random.random()
            log.debug("Produced numbers: %s" % data)
            self.emit('results', data)
            self.emit('progress', 100 * (i + 1) / self.iterations)
            sleep(self.delay)
            if self.should_stop():
                log.warning("Catch stop command in procedure")
                break
   def shutdown(self):
        log.info("Finished")
class MainWindow(ManagedWindow):
    def __init__(self):
        super(MainWindow, self).__init__(
            procedure_class=TestProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            x_axis='Iteration',
            y_axis='Random Number'
```

(continues on next page)

(continued from previous page)

```
self.setWindowTitle('GUI Example')

if __name__ == "__main__":
    if len(sys.argv) > 1:
        # If any parameter is passed, the console mode is run
        # This criteria can be changed at user discretion
        app = ManagedConsole(procedure_class=TestProcedure)

else:
    app = QtWidgets.QApplication(sys.argv)
    window = MainWindow()
    window.show()

sys.exit(app.exec())
```

If we run the script above without any parameter, you will have the graphical user interface example. If you run as follow, you will use the command line mode:

```
python console.py --iterations 10 --result-file console_test
```

Console output is as follow (to show the progress bar, you need to install the optional module progressbar2):

```
C:\GIT\pymeasurest\examples\Basic>py -3 console.py --iterations 10 --result-file console_test
03:26:29 PM: Worker thread started (root, INFO)
03:26:29 PM: Worker started running an instance of 'TestProcedure' (root, INFO)
03:26:29 PM: Setting up random number generator (root, INFO)
03:26:29 PM: Starting to generate numbers (root, INFO)
03:26:31 PM: Finished (root, INFO)
03:26:31 PM: Monitor caught stop command (pymeasure.display.listeners, INFO)
C:\GIT\pymeasurest\examples\Basic>
```

# PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20250827

## Other useful commands

To show all the command line switches:

python console.py --help

To run an experiment with parameters retrieved from an existing result file.

 $python\ console.py\ --use-result-file\ console\_test2023-08-09\_1.csv$ 

46 Chapter 3. Tutorials

# **PYMEASURE.ADAPTERS**

The adapter classes allow the instruments to be independent of the communication method used. The instrument implementation takes care of any potential quirks in its communication protocol (see *Advanced communication protocols*), and the adapter takes care of the details of the over-the-wire communication with the hardware device. In the vast majority of cases, it will be sufficient to pass a connection string or integer to the instrument (see *Connecting to an instrument*), which uses the *pymeasure.adapters.VISAAdapter* in the background.

# 4.1 Adapter base class

class pymeasure.adapters.Adapter(log=None, \*\*kwargs)

Base class for Adapter child classes, which adapt between the Instrument object and the connection, to allow flexible use of different connection techniques.

This class should only be inherited from.

## **Parameters**

- log Parent logger of the 'Adapter' logger.
- \*\*kwargs Keyword arguments just to be cooperative.

## close()

Close the connection.

## flush\_read\_buffer()

Flush and discard the input buffer. Implement in subclass.

```
read(**kwargs)
```

Read up to (excluding) read\_termination or the whole read buffer.

Do not override in a subclass!

## **Parameters**

\*\*kwargs – Keyword arguments for the connection itself.

#### Returns str

ASCII response of the instrument (excluding read termination).

**read\_binary\_values**(header\_bytes=0, termination\_bytes=None, dtype=<class 'numpy.float32'>, sep=", \*\*kwargs)

Returns a numpy array from a query for binary data

## **Parameters**

• header\_bytes (int) – Number of bytes to ignore in header.

- **termination\_bytes** (*int*) Number of bytes to strip at end of message or None.
- **dtype** The NumPy data type to format the values with.
- **sep** (*string*) Separator between chars. If given, use fromstring, otherwise frombytes.
- \*\*kwargs Further arguments for the NumPy fromstring / frombytes method.

#### Returns

NumPy array of values

#### Raises

**ValueError** – if the data buffer is empty or malformed

```
read_bytes(count=-1, break_on_termchar=False, **kwargs)
```

Read a certain number of bytes from the instrument.

Do not override in a subclass!

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read from the whole read buffer.
- **break\_on\_termchar** (*boo1*) Stop reading at a termination character.
- \*\*kwargs Keyword arguments for the connection itself.

### **Returns bytes**

Bytes response of the instrument (including termination).

## write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

Do not override in a subclass!

## **Parameters**

- **command** (str) Command string to be sent to the instrument (without termination).
- **\*\*kwargs** Keyword arguments for the connection itself.

#### write\_binary\_values(command, values, termination=", \*\*kwargs)

Write binary data to the instrument, e.g. waveform for signal generators

## **Parameters**

- **command** command string to be sent to the instrument
- **values** iterable representing the binary values
- **termination** String added afterwards to terminate the message.
- \*\*kwargs Key-word arguments to pass onto Adapter.\_format\_binary\_values()

### Returns

number of bytes written

## write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

Do not override in a subclass!

#### **Parameters**

- **content** (*bytes*) The bytes to write to the instrument.
- **\*\*kwargs** Keyword arguments for the connection itself.

# 4.2 VISA adapter

class pymeasure.adapters.VISAAdapter(resource\_name, visa\_library=", log=None, \*\*kwargs)

Bases: Adapter

Adapter class for the VISA library, using PyVISA to communicate with instruments.

The workhorse of our library, used by most instruments.

#### **Parameters**

- resource\_name A VISA resource string or GPIB address integer that identifies the target of the connection
- visa\_library PyVISA VisaLibrary Instance, path of the VISA library or VisaLibrary spec string (@py or @ivi). If not given, the default for the platform will be used.
- log Parent logger of the 'Adapter' logger.
- \*\*kwargs Keyword arguments for configuring the PyVISA connection.

#### **Kwargs**

Keyword arguments are used to configure the connection created by PyVISA. This is complicated by the fact that which arguments are valid depends on the interface (e.g. serial, GPIB, TCPI/IP, USB) determined by the current resource\_name.

A flexible process is used to easily define reasonable default values for different instrument interfaces, but also enable the instrument user to override any setting if their situation demands

A kwarg that names a pyVISA interface type (most commonly asrl, gpib, tcpip, or usb) is a dictionary with keyword arguments defining defaults specific to that interface. Example: asrl={'baud\_rate': 4200}.

All other kwargs are either generally valid (e.g. timeout=500) or override any default settings from the interface-specific entries above. For example, passing baud\_rate=115200 when connecting via a resource name ASRL1 would override a default of 4200 defined as above.

See Modifying connection settings for how to tweak settings when connecting to an instrument. See Defining default connection settings for how to best define default settings when implementing an instrument.

## close()

Close the connection.



## 1 Note

This closes the connection to the resource for all adapters using it currently (e.g. different adapters using the same GPIB line).

# flush\_read\_buffer()

Flush and discard the input buffer

As detailed by pyvisa, discard the read and receivee buffer contents and if data was present in the read buffer and no END-indicator was present, read from the device until encountering an END indicator (which causes loss of data).

4.2. VISA adapter 49

## read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

Do not override in a subclass!

#### **Parameters**

\*\*kwargs – Keyword arguments for the connection itself.

#### Returns str

ASCII response of the instrument (excluding read termination).

**read\_binary\_values**(header\_bytes=0, termination\_bytes=None, dtype=<class 'numpy.float32'>, sep=", \*\*kwargs)

Returns a numpy array from a query for binary data

#### **Parameters**

- header\_bytes (int) Number of bytes to ignore in header.
- **termination\_bytes** (*int*) Number of bytes to strip at end of message or None.
- **dtype** The NumPy data type to format the values with.
- **sep** (*string*) Separator between chars. If given, use fromstring, otherwise frombytes.
- \*\*kwargs Further arguments for the NumPy fromstring / frombytes method.

#### Returns

NumPy array of values

#### Raises

**ValueError** – if the data buffer is empty or malformed

read\_bytes(count=-1, break on termchar=False, \*\*kwargs)

Read a certain number of bytes from the instrument.

Do not override in a subclass!

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read from the whole read buffer.
- **break\_on\_termchar** (*boo1*) Stop reading at a termination character.
- \*\*kwargs Keyword arguments for the connection itself.

## **Returns bytes**

Bytes response of the instrument (including termination).

## wait\_for\_srq(timeout=25, delay=0.1)

Block until a SRQ, and leave the bit high

## **Parameters**

- timeout Timeout duration in seconds
- delay Time delay between checking SRQ in seconds

## write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

Do not override in a subclass!

## **Parameters**

- **command** (str) Command string to be sent to the instrument (without termination).
- \*\*kwargs Keyword arguments for the connection itself.

write\_binary\_values(command, values, termination=", \*\*kwargs)

Write binary data to the instrument, e.g. waveform for signal generators

#### **Parameters**

- command command string to be sent to the instrument
- **values** iterable representing the binary values
- **termination** String added afterwards to terminate the message.
- \*\*kwargs Key-word arguments to pass onto Adapter.\_format\_binary\_values()

#### Returns

number of bytes written

write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

Do not override in a subclass!

#### **Parameters**

- **content** (*bytes*) The bytes to write to the instrument.
- **\*\*kwargs** Keyword arguments for the connection itself.

# 4.3 Serial adapter

class pymeasure.adapters.SerialAdapter(port, write\_termination=", read\_termination=", \*\*kwargs)

Bases: Adapter

Adapter class for using the Python Serial package to allow serial communication to instrument

#### **Parameters**

- port Serial port
- write\_termination String appended to messages before writing them.
- **read\_termination** String expected at end of read message and removed.
- \*\*kwargs Any valid key-word argument for serial.Serial

**\_format\_binary\_values**(values, datatype='f', is\_big\_endian=False, header\_fmt='ieee')

Format values in binary format, used internally in Adapter.write\_binary\_values().

## **Parameters**

- values data to be written to the device.
- **datatype** the format string for a single element. See struct module.
- is\_big\_endian boolean indicating endianness.
- header\_fmt Format of the header prefixing the data ("ieee", "hp", "empty").

## Returns

binary string.

4.3. Serial adapter 51

#### Return type

bytes

#### close()

Close the connection.

#### flush\_read\_buffer()

Flush and discard the input buffer.

## read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

Do not override in a subclass!

#### **Parameters**

\*\*kwargs – Keyword arguments for the connection itself.

#### Returns str

ASCII response of the instrument (excluding read\_termination).

**read\_binary\_values**(header\_bytes=0, termination\_bytes=None, dtype=<class 'numpy.float32'>, sep=", \*\*kwargs)

Returns a numpy array from a query for binary data

#### **Parameters**

- header\_bytes (int) Number of bytes to ignore in header.
- **termination\_bytes** (*int*) Number of bytes to strip at end of message or None.
- **dtype** The NumPy data type to format the values with.
- **sep** (*string*) Separator between chars. If given, use from string, otherwise from bytes.
- \*\*kwargs Further arguments for the NumPy fromstring / frombytes method.

## Returns

NumPy array of values

#### Raises

**ValueError** – if the data buffer is empty or malformed

read\_bytes(count=-1, break\_on\_termchar=False, \*\*kwargs)

Read a certain number of bytes from the instrument.

Do not override in a subclass!

### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read from the whole read buffer.
- **break\_on\_termchar** (*boo1*) Stop reading at a termination character.
- \*\*kwargs Keyword arguments for the connection itself.

## **Returns bytes**

Bytes response of the instrument (including termination).

#### write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

Do not override in a subclass!

## **Parameters**

- **command** (str) Command string to be sent to the instrument (without termination).
- \*\*kwargs Keyword arguments for the connection itself.

write\_binary\_values(command, values, termination=", \*\*kwargs)

Write binary data to the instrument, e.g. waveform for signal generators

#### **Parameters**

- command command string to be sent to the instrument
- **values** iterable representing the binary values
- **termination** String added afterwards to terminate the message.
- \*\*kwargs Key-word arguments to pass onto Adapter.\_format\_binary\_values()

#### Returns

number of bytes written

write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

Do not override in a subclass!

#### **Parameters**

- **content** (*bytes*) The bytes to write to the instrument.
- \*\*kwargs Keyword arguments for the connection itself.

# 4.4 Prologix adapter

Bases: VISAAdapter

Encapsulates the additional commands necessary to communicate over a Prologix GPIB-USB Adapter, using the *VISAAdapter*.

Each PrologixAdapter is constructed based on a connection to the Prologix device itself and the GPIB address of the instrument to be communicated to. Connection sharing is achieved by using the <code>gpib()</code> method to spawn new PrologixAdapters for different GPIB addresses.

## **Parameters**

- **resource\_name** A VISA resource string that identifies the connection to the Prologix device itself, for example "ASRL5" for the 5th COM port.
- address Integer GPIB address of the desired instrument.
- auto Enable or disable read-after-write and address instrument to listen.
- **eoi** Enable or disable EOI assertion.
- eos Set command termination string (CR+LF, CR, LF, or "")
- **gpib\_read\_timeout** Set read timeout for GPIB communication in milliseconds from 1..3000
- **kwargs** Key-word arguments if constructing a new serial object

#### **Variables**

**address** – Integer GPIB address of the desired instrument.

Usage example:

```
adapter = PrologixAdapter("ASRL5::INSTR", 7)
sourcemeter = Keithley2400(adapter) # at GPIB address 7
# generate another instance with a different GPIB address:
adapter2 = adapter.gpib(9)
multimeter = Keithley2000(adapter2) # at GPIB address 9
```

To allow user access to the Prologix adapter in Linux, create the file: /etc/udev/rules.d/51-prologix.rules, with contents:

```
SUBSYSTEMS=="usb",ATTRS{idVendor}=="0403",ATTRS{idProduct}=="6001",MODE="0666"
```

Then reload the udev rules with:

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

Since the Prologix adapter uses the same communication channel (an USB CDC) for both, communication with the adapter itself, as well as communication over GPIB, certain things need to be kept in mind:

- Operations that need to read from GPIB use the standard *read()* method.
- Operations that just read responses from the Prologix itself need to add the parameter prologix=True to read(); this avoids requesting data from GPIB. This is also necessary when the adapter is put into "listen-only" mode, where all GPIB traffic is automatically being passed up.
- Binary data must be passed to the bus using <code>write\_binary\_values()</code>. This takes care of properly escaping those binary values that would otherwise be interpreted by the Prologix adapter. Note that the default for <code>write\_binary\_values()</code> are to assume floating-point binary data, and prepend IEEE headers. In order to pass just plain bytes to the adapter, tune the <code>datatype</code> and <code>header\_fmt</code> parameters:

```
multimeter.write_binary_values('W', [addr], datatype='B', header_fmt='empty')
```

**\_format\_binary\_values**(values, datatype='f', is\_big\_endian=False, header\_fmt='ieee')

Format values in binary format, used internally in write\_binary\_values().

#### **Parameters**

- **values** data to be written to the device.
- **datatype** the format string for a single element. See struct module.
- **is\_big\_endian** boolean indicating endianess.
- header\_fmt Format of the header prefixing the data ("ieee", "hp", "empty").

## Returns

binary string.

#### Return type

bytes

## property auto

Control whether to address instruments to talk after sending them a command (bool).

Configure Prologix GPIB controller to automatically address instruments to talk after sending them a command in order to read their response. The feature called, Read-After-Write, saves the user from having to issue read commands repeatedly. This property enables (True) or disables (False) this feature.

### close()

Close the connection.



#### 1 Note

This closes the connection to the resource for all adapters using it currently (e.g. different adapters using the same GPIB line).

## property eoi

Control whether to assert the EOI signal with the last character of any command sent over GPIB port (bool).

Some instruments require EOI signal to be asserted in order to properly detect the end of a command.

### property eos

Control GPIB termination characters (str).

#### possible values:

- CR+LF
- CR
- LF
- · empty string

When data from host is received, all non-escaped LF, CR and ESC characters are removed and GPIB terminators, as specified by this command, are appended before sending the data to instruments. This command does not affect data from instruments received over GPIB port.

#### flush\_read\_buffer()

Flush and discard the input buffer

As detailed by pyvisa, discard the read and receivee buffer contents and if data was present in the read buffer and no END-indicator was present, read from the device until encountering an END indicator (which causes loss of data).

## gpib(address, \*\*kwargs)

Return a Prologix Adapter object that references the GPIB address specified, while sharing the Serial connection with other calls of this function

## **Parameters**

- address Integer GPIB address of the desired instrument
- **kwargs** Arguments for the initialization

#### Returns

PrologixAdapter for specific GPIB address

## property gpib\_read\_timeout

Control the timeout value for the GPIB communication in milliseconds

possible values: 1 - 3000

## read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

Do not override in a subclass!

#### **Parameters**

\*\*kwargs – Keyword arguments for the connection itself.

#### Returns str

ASCII response of the instrument (excluding read termination).

**read\_binary\_values**(header\_bytes=0, termination\_bytes=None, dtype=<class 'numpy.float32'>, sep=", \*\*kwargs)

Returns a numpy array from a query for binary data

#### **Parameters**

- header\_bytes (int) Number of bytes to ignore in header.
- **termination\_bytes** (*int*) Number of bytes to strip at end of message or None.
- **dtype** The NumPy data type to format the values with.
- **sep** (*string*) Separator between chars. If given, use fromstring, otherwise frombytes.
- \*\*kwargs Further arguments for the NumPy fromstring / frombytes method.

#### Returns

NumPy array of values

#### Raises

**ValueError** – if the data buffer is empty or malformed

read\_bytes(count=-1, break on termchar=False, \*\*kwargs)

Read a certain number of bytes from the instrument.

Do not override in a subclass!

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read from the whole read buffer.
- **break\_on\_termchar** (*bool*) Stop reading at a termination character.
- \*\*kwargs Keyword arguments for the connection itself.

## **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Perform a power-on reset of the controller.

The process takes about 5 seconds. All input received during this time is ignored and the connection is closed.

## property version

Get the version string of the Prologix controller.

#### wait\_for\_srq(timeout=25, delay=0.1)

Blocks until a SRQ, and leaves the bit high

## **Parameters**

• timeout – Timeout duration in seconds.

• **delay** – Time delay between checking SRQ in seconds.

#### Raises

**TimeoutError** – "Waiting for SRQ timed out."

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

If the GPIB address in address is defined, it is sent first.

#### **Parameters**

- **command** (str) Command string to be sent to the instrument (without termination).
- **kwargs** Keyword arguments for the connection itself.

```
write_binary_values(command, values, **kwargs)
```

Write binary data to the instrument, e.g. waveform for signal generators.

values are encoded in a binary format according to IEEE 488.2 Definite Length Arbitrary Block Response Data block.

#### **Parameters**

- command SCPI command to be sent to the instrument
- values iterable representing the binary values
- **kwargs** Key-word arguments to pass onto \_format\_binary\_values()

#### Returns

number of bytes written

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

Do not override in a subclass!

#### **Parameters**

- **content** (*bytes*) The bytes to write to the instrument.
- \*\*kwargs Keyword arguments for the connection itself.

# 4.5 Test adapters

These pieces are useful when writing tests.

Context manager that checks sent/received instrument commands without a device connected.

Given an instrument class and a list of command-response pairs, this context manager confirms that the code in the context manager block produces the expected messages.

Terminators are excluded from the protocol definition, as those are typically a detail of the communication method (i.e. Adapter), and not the protocol itself.

#### **Parameters**

4.5. Test adapters 57

- instrument\_cls (pymeasure.Instrument) Instrument subclass to instantiate.
- **comm\_pairs** (list[2-tuples[str]]) List of command-response pairs, i.e. 2-tuples like ('VOLT?', '3.14'). 'None' indicates that a pair member (command or response) does not exist, e.g. (None, 'RESP1'). Commands and responses are without termination characters.
- **connection\_attributes** Dictionary of connection attributes and their values.
- connection\_methods Dictionary of method names of the connection and their return values.
- \*\*kwargs Keyword arguments for the instantiation of the instrument.

Bases: Adapter

Adapter class for testing the command exchange protocol without instrument hardware.

This adapter is primarily meant for use within pymeasure.test.expected\_protocol().

The connection attribute is a unittest.mock.MagicMock such that every call returns. If you want to set a return value, you can use adapter.connection.some\_method.return\_value = 7, such that a call to adapter.connection.some\_method() will return 7. Similarly, you can verify that this call to the connection method happened with assert adapter.connection.some\_method.called is True. You can specify dictionaries with return values of attributes and methods.

#### **Parameters**

- **comm\_pairs** (*list*) List of "reference" message pair tuples. The first element is what is sent to the instrument, the second one is the returned message. 'None' indicates that a pair member (write or read) does not exist. The messages do **not** include the termination characters.
- **connection\_attributes** Dictionary of connection attributes and their values.
- connection\_methods Dictionary of method names of the connection and their return values.

## flush\_read\_buffer()

Flush and discard the input buffer

As detailed by pyvisa, discard the read buffer contents and if data was present in the read buffer and no END-indicator was present, read from the device until encountering an END indicator (which causes loss of data).

class pymeasure.adapters.FakeAdapter(log=None, \*\*kwargs)

Bases: Adapter

Provides a fake adapter for debugging purposes, which bounces back the command so that arbitrary values testing is possible.

```
a = FakeAdapter()
assert a.read() == ""
a.write("5")
assert a.read() == "5"
assert a.read() == ""
```

(continues on next page)

(continued from previous page)

```
assert a.ask("10") == "10"
assert a.values("10") == [10]
```

#### close()

Close the connection.

#### flush\_read\_buffer()

Flush and discard the input buffer. Implement in subclass.

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

Read up to (excluding) read\_termination or the whole read buffer.

Do not override in a subclass!

#### **Parameters**

\*\*kwargs – Keyword arguments for the connection itself.

#### Returns str

ASCII response of the instrument (excluding read\_termination).

```
read_binary_values(header_bytes=0, termination_bytes=None, dtype=<class 'numpy.float32'>, sep=", **kwargs)
```

Returns a numpy array from a query for binary data

#### **Parameters**

- header\_bytes (int) Number of bytes to ignore in header.
- **termination\_bytes** (*int*) Number of bytes to strip at end of message or None.
- **dtype** The NumPy data type to format the values with.
- **sep** (*string*) Separator between chars. If given, use fromstring, otherwise frombytes.
- \*\*kwargs Further arguments for the NumPy fromstring / frombytes method.

#### Returns

NumPy array of values

## Raises

**ValueError** – if the data buffer is empty or malformed

```
read_bytes(count=-1, break_on_termchar=False, **kwargs)
```

Read a certain number of bytes from the instrument.

Do not override in a subclass!

## **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read from the whole read buffer.
- **break\_on\_termchar** (*bool*) Stop reading at a termination character.
- \*\*kwargs Keyword arguments for the connection itself.

#### **Returns bytes**

Bytes response of the instrument (including termination).

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

Do not override in a subclass!

4.5. Test adapters 59

#### **Parameters**

- **command** (*str*) Command string to be sent to the instrument (without termination).
- \*\*kwargs Keyword arguments for the connection itself.

write\_binary\_values(command, values, termination=", \*\*kwargs)

Write binary data to the instrument, e.g. waveform for signal generators

#### **Parameters**

- **command** command string to be sent to the instrument
- **values** iterable representing the binary values
- **termination** String added afterwards to terminate the message.
- \*\*kwargs Key-word arguments to pass onto Adapter.\_format\_binary\_values()

## Returns

number of bytes written

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

Do not override in a subclass!

#### **Parameters**

- **content** (*bytes*) The bytes to write to the instrument.
- **\*\*kwargs** Keyword arguments for the connection itself.

## class pymeasure.generator.Generator

Generates tests from the communication with an instrument.

Example usage:

```
g = Generator()
inst = g.instantiate(TC038, "COM5", 'hcp', adapter_kwargs={'baud_rate': 9600})
inst.information # returns the 'information' property and adds it to the tests
inst.setpoint = 20
inst.setpoint == 20 # should be True
g.write_file("test_tc038.py") # write the tests to a file
```

instantiate(instrument\_class, adapter, manufacturer, adapter\_kwargs=None, \*\*kwargs)

Instantiate the instrument and store the instantiation communication.

..note:

```
You have to give all keyword arguments necessary for adapter instantiation in `adapter_kwargs`, even those, which are defined somewhere in the instrument's ``__init__`` method, be it as a default value, be it directly in the ``Instrument.__init__()`` call.
```

#### **Parameters**

- instrument\_class Class of the instrument to test.
- **adapter** Adapter (instance or str) for the instrument instantiation.
- manufacturer Module from which to import the instrument, e.g. 'hcp' if instrument\_class is 'pymeasure.hcp.tc038'.

- adapter\_kwargs Keyword arguments for the adapter instantiation (see note above).
- **\*\*kwargs** Keyword arguments for the instrument instantiation.

#### Returns

A man-in-the-middle instrument, which can be used like a normal instrument.

## parse\_stream()

Parse the stream not yet read.

#### test\_method(method name, \*args, \*\*kwargs)

Test calling the *method\_name* of the instruments with *args* and *kwargs*.

## test\_property\_getter(property)

Test getting the *property* of the instrument, adding it to the list.

## test\_property\_setter(property, value)

Test setting the *property* of the instrument to *value*, adding it to the list.

## test\_property\_setter\_batch(property, values)

Test setting *property* to each element in *values*.

## write\_file(filename='tests.py')

Write the tests into the file.

#### **Parameters**

**filename** – Name to save the tests to, may contain the path, e.g. "/tests/test\_abc.py".

## write\_getter\_test(file, property, parameters)

Write a getter test.

## write\_init\_test(file)

Write the header and init test.

## write\_method\_test(file, method, parameters)

Write a test for a method.

## write\_method\_tests(file)

Write all parametrized method tests in alphabetic order.

## write\_property\_tests(file)

Write tests for properties in alphabetic order.

If getter and setter exist, the setter is the first test.

## write\_setter\_test(file, property, parameters)

Write a setter test.

4.5. Test adapters 61

PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20250827			
,	3		

# **PYMEASURE.EXPERIMENT**

This section contains specific documentation on the classes and methods of the package.

# 5.1 Experiment class

The Experiment class is intended for use in the Jupyter notebook environment.

Bases: object

Class which starts logging and creates/runs the results and worker processes.

```
procedure = Procedure()
experiment = Experiment(title, procedure)
experiment.start()
experiment.plot_live('x', 'y', style='.-')

for a multi-subplot graph:

import pylab as pl
ax1 = pl.subplot(121)
experiment.plot('x','y',ax=ax1)
ax2 = pl.subplot(122)
experiment.plot('x','z',ax=ax2)
experiment.plot_live()
```

## Variables

**value** – The value of the parameter

## **Parameters**

- **title** The experiment title
- **procedure** The procedure object
- analyse Post-analysis function, which takes a pandas dataframe as input and returns it with added (analysed) columns. The analysed results are accessible via experiment.data, as opposed to experiment.results.data for the 'raw' data.
- \_data\_timeout Time limit for how long live plotting should wait for datapoints.

## clear\_plot()

Clear the figures and plot lists.

## property data

Data property which returns analysed data, if an analyse function is defined, otherwise returns the raw data.

```
plot(*args, **kwargs)
```

Plot the results from the experiment.data pandas dataframe. Store the plots in a plots list attribute.

```
plot_live(*args, **kwargs)
```

Live plotting loop for jupyter notebook, which automatically updates (an) in-line matplotlib graph(s). Will create a new plot as specified by input arguments, or will update (an) existing plot(s).

#### start()

Start the worker

```
update_line(ax, hl, xname, yname)
```

Update a line in a matplotlib graph with new data.

## update\_plot()

Update the plots in the plots list with new data from the experiment.data pandas dataframe.

#### wait\_for\_data()

Wait for the data attribute to fill with datapoints.

```
pymeasure.experiment.experiment.create_filename(title)
```

Create a new filename according to the style defined in the config file. If no config is specified, create a temporary file.

```
pymeasure.experiment.experiment.get_array(start, stop, step)
```

Returns a numpy array from start to stop

```
pymeasure.experiment.get_array_steps(start, stop, numsteps)
```

Returns a numpy array from start to stop in numsteps

```
pymeasure.experiment.get_array_zero(maxval, step)
```

Returns a numpy array from 0 to maxval to -maxval to 0

## 5.2 Listener class

```
class pymeasure.experiment.listeners.Listener(port, topic=", timeout=0.01)
```

Bases: StoppableThread

Base class for Threads that need to listen for messages on a ZMQ TCP port and can be stopped by a thread-safe method call

## message\_waiting()

Check if we have a message, wait at most until timeout.

```
receive(flags=0)
```

```
class pymeasure.experiment.listeners.Monitor(results, queue)
```

Bases: QueueListener

```
class pymeasure.experiment.listeners.Recorder(results, queue, **kwargs)
```

```
Bases: QueueListener
```

Recorder loads the initial Results for a filepath and appends data by listening for it over a queue. The queue ensures that no data is lost between the Recorder and Worker.

#### stop()

Stop the listener.

This asks the thread to terminate, and then waits for it to do so. Note that if you don't call this before your application exits, there may be some records still left on the queue, which won't be processed.

## 5.3 Procedure class

#### class pymeasure.experiment.procedure.Procedure(\*\*kwargs)

Provides the base class of a procedure to organize the experiment execution. Procedures should be run by Workers to ensure that asynchronous execution is properly managed.

```
procedure = Procedure()
results = Results(procedure, data_filename)
worker = Worker(results, port)
worker.start()
```

Inheriting classes should define the startup, execute, and shutdown methods as needed. The shutdown method is called even with a software exception or abort event during the execute method.

If keyword arguments are provided, they are added to the object as attributes.

## check\_parameters()

Raises an exception if any parameter is missing before calling the associated function. Ensures that each value can be set and got, which should cast it into the right format. Used as a decorator @check\_parameters on the startup method

## evaluate\_metadata()

Evaluates all Metadata objects, fixing their values to the current value

## execute()

Preforms the commands needed for the measurement itself. During execution the shutdown method will always be run following this method. This includes when Exceptions are raised.

#### gen\_measurement()

Create MEASURE and DATA\_COLUMNS variables for get\_datapoint method.

## get\_estimates()

Function that returns estimates that are to be displayed by the EstimatorWidget. Must be reimplemented by subclasses. Should return an int or float representing the duration in seconds, or a list with a tuple for each estimate. The tuple should consists of two strings: the first will be used as the label of the estimate, the second as the displayed estimate.

## $is_last() \rightarrow bool$

Check if the procedure is the last one in the queue.

This method must be monkey patched by a worker to provide functionality.

#### **Returns:**

bool: True if the procedure is the last one in the queue, False otherwise.

5.3. Procedure class 65

## metadata\_objects()

Returns a dictionary of all the Metadata objects

## parameter\_objects()

Returns a dictionary of all the Parameter objects and grabs any current values that are not in the default definitions

#### parameter\_values()

Returns a dictionary of all the Parameter values and grabs any current values that are not in the default definitions

#### parameters\_are\_set()

Returns True if all parameters are set

## static parse\_columns(columns)

Get columns with any units in parentheses. For each column, if there are matching parentheses containing text with no spaces, parse the value between the parentheses as a Pint unit. For example, "Source Voltage (V)" will be parsed and matched to Unit('volt'). Raises an error if a parsed value is undefined in Pint unit registry. Return a dictionary of matched columns with their units.

#### **Parameters**

**columns** – List of columns to be parsed.

#### Returns

Dictionary of columns with Pint units.

## classmethod placeholder\_names()

Collect the names of all eligible placeholders (parameters & metadata)

## placeholder\_objects()

Collect all eligible placeholders (parameters & metadata) with their value in a dict.

## refresh\_parameters()

Enforces that all the parameters are re-cast and updated in the meta dictionary

#### set\_parameters(parameters, except\_missing=True)

Sets a dictionary of parameters and raises an exception if additional parameters are present if except\_missing is True

## shutdown()

Executes the commands necessary to shut down the instruments and leave them in a safe state. This method is always run at the end.

#### startup()

Executes the commands needed at the start-up of the measurement

## class pymeasure.experiment.procedure.UnknownProcedure(parameters)

Handles the case when a Procedure object can not be imported during loading in the Results class

## startup()

Executes the commands needed at the start-up of the measurement

# 5.4 Parameter classes

The parameter classes are used to define input variables for a *Procedure*. They each inherit from the *Parameter* base class.

Parameter sub-class that uses the boolean type to store the value.

#### **Variables**

value – The boolean value of the parameter

#### **Parameters**

- name The parameter name
- default The default boolean value
- ui\_class A Qt class to use for the UI of this parameter

## convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

#### **Parameters**

value - value to be converted

#### **Returns**

converted value

Parameter sub-class that uses the floating point type to store the value.

#### **Variables**

**value** – The floating point value of the parameter

#### **Parameters**

- **name** The parameter name
- units The units of measure for the parameter
- minimum The minimum allowed value (default: -1e9)
- maximum The maximum allowed value (default: 1e9)
- **decimals** The number of decimals considered (default: 15)
- default The default floating point value
- ui\_class A Qt class to use for the UI of this parameter
- step step size for parameter's UI spinbox. If None, spinbox will have step disabled

## convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

5.4. Parameter classes 67

#### **Parameters**

value - value to be converted

#### Returns

converted value

class pymeasure.experiment.parameters.IntegerParameter(name, units=None,

minimum=-1000000000.0, maximum=1000000000.0, step=None, \*\*kwargs)

Parameter sub-class that uses the integer type to store the value.

#### **Variables**

value - The integer value of the parameter

#### **Parameters**

- name The parameter name
- units The units of measure for the parameter
- minimum The minimum allowed value (default: -1e9)
- maximum The maximum allowed value (default: 1e9)
- **default** The default integer value
- ui\_class A Qt class to use for the UI of this parameter
- step int step size for parameter's UI spinbox. If None, spinbox will have step disabled

## convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

## **Parameters**

value - value to be converted

#### Returns

converted value

**class** pymeasure.experiment.parameters.**ListParameter**(name, choices=None, units=None, \*\*kwargs)

\*Parameter sub-class that stores the value as a list. String representation of choices must be unique.

## Parameters

- **name** The parameter name
- choices An explicit list of choices, which is disregarded if None
- units The units of measure for the parameter
- default The default value
- ui\_class A Qt class to use for the UI of this parameter

# property choices

Returns an immutable iterable of choices, or None if not set.

## convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

#### **Parameters**

value - value to be converted

#### Returns

converted value

Encapsulates the information for a measurable experiment parameter with information about the name, fget function and units if supplied. The value property is called when the procedure retrieves a datapoint and calls the fget function. If no fget function is specified, the value property will return the latest set value of the parameter (or default if never set).

#### **Variables**

**value** – The value of the parameter

#### **Parameters**

- name The parameter name
- **fget** The parameter fget function (e.g. an instrument parameter)
- default The default value

class pymeasure.experiment.parameters.Metadata(name, fget=None, units=None, default=None, fmt='%s')

Encapsulates the information for metadata of the experiment with information about the name, the fget function and the units, if supplied. If no fget function is specified, the value property will return the latest set value of the parameter (or default if never set).

## **Variables**

**value** – The value of the parameter. This returns (if a value is set) the value obtained from the *fget* (after evaluation) or a manually set value. Returns *None* if no value has been set

#### **Parameters**

- name The parameter name
- **fget** The parameter fget function; can be provided as a callable, or as a string, in which case it is assumed to be the name of a method or attribute of the *Procedure* class in which the Metadata is defined. Passing a string also allows for nested attributes by separating them with a period (e.g. to access an attribute or method of an instrument) where only the last attribute can be a method.
- **units** The parameter units
- default The default value, in case no value is assigned or if no fget method is provided
- fmt A string used to format the value upon writing it to a file. Default is "%s"

is\_set()

Returns True if the Parameter value is set

Encapsulates the information for an experiment parameter with information about the name, and units if supplied.

## Variables

value - The value of the parameter

#### **Parameters**

5.4. Parameter classes 69

- **name** The parameter name
- default The default value
- ui\_class A Qt class to use for the UI of this parameter
- **group\_by** Defines the Parameter(s) that controls the visibility of the associated input; can be a string containing the Parameter name, a list of strings with multiple Parameter names, or a dict containing {"Parameter name": condition} pairs.
- **group\_condition** The condition for the group\_by Parameter that controls the visibility of this parameter, provided as a value or a (lambda)function. If the group\_by argument is provided as a list of strings, this argument can be either a single condition or a list of conditions. If the group\_by argument is provided as a dict this argument is ignored.

### Description

A string providing a human-friendly description for the parameter.

# property cli\_args

helper for command line interface parsing of parameters

This property returns a list of data to help formatting a command line interface interpreter, the list is composed of the following elements: - index 0: default value - index 1: List of value to format an help string, that is either, the name of the fields to be documented or a tuple with (helps\_string, field) - index 2: type

#### convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

#### **Parameters**

value - value to be converted

## Returns

converted value

#### is\_set()

Returns True if the Parameter value is set

*VectorParameter* sub-class of 2 dimensions to store a value and its uncertainty.

## Variables

value – The value of the parameter as a list of 2 floating point numbers

#### **Parameters**

- **name** The parameter name
- uncertainty\_type Type of uncertainty, 'absolute', 'relative' or 'percentage'
- units The units of measure for the parameter
- **default** The default value
- ui\_class A Qt class to use for the UI of this parameter

#### convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

#### **Parameters**

value – value to be converted

#### Returns

converted value

class pymeasure.experiment.parameters.VectorParameter(name, length=3, units=None, \*\*kwargs)

Parameter sub-class that stores the value in a vector format.

### Variables

value – The value of the parameter as a list of floating point numbers

#### **Parameters**

- **name** The parameter name
- length The integer dimensions of the vector
- units The units of measure for the parameter
- **default** The default value
- ui\_class A Qt class to use for the UI of this parameter

#### convert(value)

Convert user input to python data format

Subclasses are expected to customize this method. Default implementation is the identity function

#### **Parameters**

value - value to be converted

#### Returns

converted value

# 5.5 Worker class

```
class pymeasure.experiment.workers.Worker(results, log_queue=None, log_level=20, port=None)
```

Bases: StoppableThread

Worker runs the procedure and emits information about the procedure and its status over a ZMQ TCP port. In a child thread, a Recorder is run to write the results to

```
emit(topic: str, record: Any)
```

Emits data of some topic over TCP

handle\_abort()

handle\_batch\_record(record: Any)

handle\_error()

handle\_record(record: dict[str, Any])

is\_last()

join(timeout: int = 0)

Joins the current thread and forces it to stop after the timeout if necessary

## **Parameters**

timeout - Timeout duration in seconds

5.5. Worker class 71

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

# shutdown()

```
update_status(status: int)
```

# 5.6 Results class

```
class pymeasure.experiment.results.CSVFormatter(columns, delimiter=',')
```

Formatter of data results

## format(record)

Formats a record as csv.

#### **Parameters**

**record** (*dict*) – record to format.

#### Returns

a string

class pymeasure.experiment.results.Results(procedure, data\_filename)

The Results class provides a convenient interface to reading and writing data in connection with a *Procedure* object.

# Variables

- **COMMENT** The character used to identify a comment (default: #)
- **DELIMITER** The character used to delimit the data (default: ,)
- LINE\_BREAK The character used for line breaks (default n)
- CHUNK\_SIZE The length of the data chuck that is read

## **Parameters**

- **procedure** Procedure object
- data\_filename The data filename where the data is or should be stored

## format(data)

Returns a formatted string containing the data to be written to a file

## header()

Returns a text header to accompany a datafile so that the procedure can be reconstructed

## labels()

Returns the columns labels as a string to be written to the file

```
static load(data_filename, procedure_class=None)
```

Returns a Results object with the associated Procedure object and data

## metadata()

Returns a text header for the metadata to write into the datafile

```
parse(line)
```

Returns a dictionary containing the data from the line

```
static parse_header(header, procedure_class=None)
```

Returns a Procedure object with the parameters as defined in the header text.

### reload()

Preforms a full reloading of the file data, neglecting any changes in the comments

#### store\_metadata()

Inserts the metadata header (if any) into the datafile

Replace placeholders in string with values from procedure parameters.

Replaces the placeholders in the provided string with the values of the associated parameters, as provided by the procedure. This uses the standard python string.format syntax. Apart from the parameter in the procedure (which should be called by their full names) "date" and "time" are also added as optional placeholders.

#### **Parameters**

- **string** The string in which the placeholders are to be replaced. Python string.format syntax is used, e.g. "{Parameter Name}" to insert a FloatParameter called "Parameter Name", or "{Parameter Name:.2f}" to also specifically format the parameter.
- **procedure** The procedure from which to get the parameter values.
- date\_format A string to represent how the additional placeholder "date" will be formatted.
- time\_format A string to represent how the additional placeholder "time" will be formatted.

```
pymeasure.experiment.results.unique_filename(directory, prefix='DATA', suffix='', ext='csv', dated_folder=False, index=True, datetimeformat='%Y-%m-%d', procedure=None)
```

Returns a unique filename based on the directory and prefix

5.6. Results class 73

PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20250827				

**CHAPTER** 

SIX

# **PYMEASURE.DISPLAY**

This section contains specific documentation on the classes and methods of the package.

# 6.1 Browser classes

class pymeasure.display.browser.BaseBrowserItem

Bases: object

Base class for an experiment's browser item. BaseBrowerItem outlines core functionality for displaying progress of an experiment to the user.

Bases: QTreeWidget

Graphical list view of *Experiment* objects allowing the user to view the status of queued Experiments as well as loading and displaying data from previous runs.

In order that different Experiments be displayed within the same Browser, they must have entries in *DATA\_COLUMNS* corresponding to the *measured\_quantities* of the Browser.

add(experiment)

Add a *Experiment* object to the Browser. This function checks to make sure that the Experiment measures the appropriate quantities to warrant its inclusion, and then adds a BrowserItem to the Browser, filling all relevant columns with Parameter data.

class pymeasure.display.browser.BrowserItem(results, color, parent=None)

Bases: QTreeWidgetItem, BaseBrowserItem

Represent a row in the Browser tree widget

# 6.2 Console class

```
class pymeasure.display.console.ConsoleArgumentParser(procedure_class, **kwargs)
```

Bases: ArgumentParser

setup\_parser()

Setup command line arguments parsing from parameters information

class pymeasure.display.console.ConsoleBrowserItem(progress\_bar)

Bases: BaseBrowserItem

Bases: QCoreApplication

Base class for console experiment management.

Parameters for \_\_init\_\_ constructor.

#### **Parameters**

- **procedure\_class** procedure class describing the experiment (see *Procedure*)
- log\_channel logging.Logger instance to use for logging output
- log\_level logging level
- kwargs additional keyword arguments to be passed to the ConsoleArgumentParser constructor

abort()

Aborts the currently running Experiment, but raises an exception if there is no running experiment

```
exec() \rightarrow int
```

get\_filename(directory, procedure=None)

Return filename for saving results file

#### **Parameters**

**directory** – directory of the returned filename.

# 6.3 Curves classes

```
class pymeasure.display.curves.BufferCurve(**kwargs)
```

Bases: PlotDataItem

Creates a curve based on a predefined buffer size and allows data to be added dynamically.

```
append(x, y)
```

Appends data to the curve with optional errors

prepare(size, dtype=<class 'numpy.float32'>)

Prepares the buffer based on its size, data type

class pymeasure.display.curves.Crosshairs(plot, pen=None)

Bases: QObject

Attaches crosshairs to the a plot and provides a signal with the x and y graph coordinates

mouseMoved(event=None)

Updates the mouse position upon mouse movement

update()

Updates the mouse position based on the data in the plot. For dynamic plots, this is called each time the data changes to ensure the x and y values correspond to those on the display.

**class** pymeasure.display.curves.**ResultsCurve**(results, x, y, force\_reload=False, wdg=None, \*\*kwargs)

Bases: PlotDataItem

Creates a curve loaded dynamically from a file through the Results object. The data can be forced to fully reload on each update, useful for cases when the data is changing across the full file instead of just appending.

## update\_data()

Updates the data by polling the results

class pymeasure.display.curves.ResultsImage(results, x, y, z, force\_reload=False, wdg=None, \*\*kwargs)

Bases: ImageItem

Creates an image loaded dynamically from a file through the Results object.

#### colormap(x)

Return mapped color as 0.0-1.0 floats RGBA

#### $find_img_index(x, y)$

Finds the integer image indices corresponding to the closest x and y points of the data given some x and y data

#### $round_up(x)$

Convenience function since numpy rounds to even

# 6.4 Inputs classes

```
class pymeasure.display.inputs.BooleanInput(parameter, parent=None, **kwargs)
```

Bases: Input, QCheckBox

Checkbox for boolean values, connected to a BooleanParameter.

## set\_parameter(parameter)

Connects a new parameter to the input box, and initializes the box value.

#### **Parameters**

**parameter** – parameter to connect.

```
class pymeasure.display.inputs.Input(parameter, **kwargs)
```

Bases: object

Mix-in class that connects a *Parameter* object to a GUI input box.

## **Parameters**

**parameter** – The parameter to connect to this input box.

## Attr parameter

Read-only property to access the associated parameter.

# property parameter

The connected parameter object. Read-only property; see set\_parameter().

Note that reading this property will have the side-effect of updating its value from the GUI input box.

# set\_parameter(parameter)

Connects a new parameter to the input box, and initializes the box value.

#### **Parameters**

**parameter** – parameter to connect.

#### update\_parameter()

Update the parameter value with the Input GUI element's current value.

6.4. Inputs classes 77

```
class pymeasure.display.inputs.IntegerInput(parameter, parent=None, **kwargs)
     Bases: Input, QSpinBox
     Spin input box for integer values, connected to a IntegerParameter.
     set_parameter(parameter)
           Connects a new parameter to the input box, and initializes the box value.
                   parameter – parameter to connect.
     stepEnabled(self) \rightarrow QAbstractSpinBox.StepEnabled
class pymeasure.display.inputs.ListInput(parameter, parent=None, **kwargs)
     Bases: Input, QComboBox
     Dropdown for list values, connected to a ListParameter.
     set_parameter(parameter)
           Connects a new parameter to the input box, and initializes the box value.
               Parameters
                   parameter – parameter to connect.
class pymeasure.display.inputs.ScientificInput(parameter, parent=None, **kwargs)
     Bases: Input, QDoubleSpinBox
     Spinner input box for floating-point values, connected to a FloatParameter. This box will display and accept
     values in scientific notation when appropriate.
       See also
       Class FloatInput
             For a non-scientific floating-point input box.
     set_parameter(parameter)
           Connects a new parameter to the input box, and initializes the box value.
               Parameters
                   parameter – parameter to connect.
     stepEnabled(self) \rightarrow QAbstractSpinBox.StepEnabled
     textFromValue(self, v: float) \rightarrow str
     validate(self, input: str | None, pos: int)
     valueFromText(self, text: str \mid None) \rightarrow float
class pymeasure.display.inputs.StringInput(parameter, parent=None, **kwargs)
     Bases: Input, QLineEdit
     String input box connected to a Parameter. Parameter subclasses that are string-based may also use this input,
```

String input box connected to a Parameter. Parameter subclasses that are string-based may also use this input, but non-string parameters should use more specialised input classes.

# 6.5 Listeners classes

```
{\bf class} \ {\bf pymeasure.display.listeners.Monitor} ({\it queue})
```

Bases: QThread

Monitor listens for status and progress messages from a Worker through a queue to ensure no messages are losts

run(self)

class pymeasure.display.listeners.QListener(port, topic=", timeout=0.01)

Bases: StoppableQThread

Base class for QThreads that need to listen for messages on a ZMQ TCP port and can be stopped by a threadand process-safe method call

# 6.6 Log classes

# class pymeasure.display.log.LogHandler

Bases: Handler

class Emitter

Bases: QObject

emit(record)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

# 6.7 Manager classes

```
class pymeasure.display.manager.BaseManager(port=5888, log_level=20, parent=None)
```

Bases: QObject

Controls the execution of *Experiment* classes by implementing a queue system in which Experiments are added, removed, executed, or aborted.

abort()

Aborts the currently running Experiment, but raises an exception if there is no running experiment

clear()

Remove all Experiments

is\_running()

Returns True if a procedure is currently running

load(experiment)

Load a previously executed Experiment

next()

Initiates the start of the next experiment in the queue as long as no other experiments are currently running and there is a procedure in the queue.

6.5. Listeners classes 79

```
queue(experiment)
```

Adds an experiment to the queue.

#### remove(experiment)

Removes an Experiment

#### resume()

Resume processing of the queue.

Bases: QObject

The Experiment class helps group the *Procedure*, *Results*, and their display functionality. Its function is only a convenient container.

#### **Parameters**

- results Results object
- **curve\_list** *ResultsCurve* list. List of curves associated with an experiment. They could represent different views of the same experiment. Not required for .*ManagedConsole* displayed experiments.
- browser\_item BaseBrowserItem based object

## class pymeasure.display.manager.ExperimentQueue

Bases: QObject

Represents a queue of Experiments and allows queries to be easily preformed.

#### has\_next()

Returns True if another item is on the queue

next()

Returns the next experiment on the queue

class pymeasure.display.manager.Manager(widget\_list, browser, port=5888, log\_level=20, parent=None)

Bases: BaseManager

Controls the execution of *Experiment* classes by implementing a queue system in which Experiments are added, removed, executed, or aborted. When instantiated, the Manager is linked to a *Browser* and a PyQtGraph *PlotItem* within the user interface, which are updated in accordance with the execution status of the Experiments.

## load(experiment)

Load a previously executed Experiment

remove(experiment)

Removes an Experiment

# 6.8 Plotter class

class pymeasure.display.plotter.Plotter(results, refresh\_time=0.1, linewidth=1)

Bases: StoppableThread

Plotter dynamically plots data from a file through the Results object.



#### **Tutorial** *Using the Plotter*

A tutorial and example on using the Plotter and PlotterWindow.

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

## setup\_plot(plot)

This method does nothing by default, but can be overridden by the child class in order to set up custom options for the plot window, via its PlotItem.

#### **Parameters**

**plot** – This window's PlotItem instance.

# 6.9 Qt classes

All Qt imports should reference pymeasure.display.Qt, for consistent importing from either PySide or PyQt4.

```
Qt.fromUi(**kwargs)
```

Returns a Qt object constructed using loadUiType based on its arguments. All QWidget objects in the form class are set in the returned object for easy accessibility.

# 6.10 Thread classes

class pymeasure.display.thread.StoppableQThread(parent=None)

Bases: QThread

Base class for QThreads which require the ability to be stopped by a thread-safe method call

join(timeout=0)

Joins the current thread and forces it to stop after the timeout if necessary

## **Parameters**

timeout - Timeout duration in seconds

6.8. Plotter class 81

# 6.11 Widget classes

class pymeasure.display.widgets.browser\_widget.BrowserWidget(\*args, parent=None)

Bases: QWidget

Widget wrapper for Browser class

class pymeasure.display.widgets.directory\_widget.DirectoryLineEdit(parent=None)

Bases: QLineEdit

Widget that allows to choose a directory path. A completer is implemented for quick completion. A browse button is available.

class pymeasure.display.widgets.estimator\_widget.EstimatorThread(get\_estimates\_callable)

Bases: StoppableQThread

run(self)

class pymeasure.display.widgets.estimator\_widget.EstimatorWidget(parent=None)

Bases: QWidget

Widget that allows to display up-front estimates of the measurement procedure.

This widget relies on a *get\_estimates* method of the *Procedure* class. *get\_estimates* is expected to return a list of tuples, where each tuple contains two strings: a label and the estimate.

If the SequencerWidget is also used, it is possible to ask for the current sequencer or its length by asking for two keyword arguments in the Implementation of the get\_estimates function: sequence and sequence\_length, respectively.

## check\_get\_estimates\_signature()

Method that checks the signature of the get\_estimates function. It checks which input arguments are allowed and, if the output is correct for the EstimatorWidget, stores the number of estimates.

## display\_estimates(estimates)

Method that updates the shown estimates for the given set of estimates.

#### **Parameters**

**estimates** – The set of estimates to be shown in the form of a list of tuples of (2) strings

## get\_estimates()

Method that makes a procedure with the currently entered parameters and returns the estimates for these parameters.

#### update\_estimates()

Method that gets and displays the estimates. Implemented for connecting to the 'update'-button.

class pymeasure.display.widgets.fileinput\_widget.FileInputWidget(parent=None)

Bases: QWidget

Widget for controlling where the data of an experiment will be stored.

The widget consists of a field for the filename (*FilenameLineEdit*), a field for the directory (*DirectoryLineEdit*), and a checkbox to control whether the measurement is stored.

## property directory

String controlling the directory where the file will be stored.

### property extensions

List of extensions that are recognized by the widget.

The first value of this list will be used as default value in case no extension is provided in the filename input field.

## property filename

String controlling the filename that is shown in the filename input field.

### property filename\_base

String containing the base of the filename with which the file will be stored. Can only be read.

## property filename\_extension

String containing the file extension with which the file will be stored.

Can only be read.

## property filename\_fixed

Boolean controlling whether the filename input field is frozen. If *True*, the filename field will be visible but disabled (i.e., grayed out).

## property store\_measurement

Boolean controlling whether the measurement will be stored.

Bases: QLineEdit

Widget that allows to choose a filename. A completer is implemented for quick completion of placeholders

```
class pymeasure.display.widgets.filename_widget.FilenameValidator(placeholders, parent)
```

Bases: QValidator

```
fixup(self, a0: str \mid None) \rightarrow str
```

```
validate(self, a0: str | None, a1: int)
```

class pymeasure.display.widgets.filename\_widget.PlaceholderCompleter(placeholders)

Bases: QCompleter

```
splitPath(self, path: str \mid None) \rightarrow list[str]
```

Bases: PlotFrame

Extends PlotFrame to plot also axis Z using colors

#### **ResultsClass**

alias of ResultsImage

```
class pymeasure.display.widgets.image_widget.ImageWidget(name, columns, x_axis, y_axis, z_axis=None, refresh_time=0.2, check_status=True, parent=None)
```

Bases: TabWidget, QWidget

Extends the ImageFrame to allow different columns of the data to be dynamically chosen

```
load(curve)
           Add curve to widget
     new_curve(results, color=<PyQt5.QtGui.QColor object>, **kwargs)
           Creates a new image
     remove(curve)
           Remove curve from widget
     sizeHint(self) \rightarrow QSize
class pymeasure.display.widgets.inputs_widget.InputsWidget(procedure_class, inputs=(),
                                                                       parent=None, hide_groups=True,
                                                                       inputs_in_scrollarea=False)
     Bases: QWidget
     Widget wrapper for various Inputs classes
     get_procedure()
           Returns the current procedure
class pymeasure.display.widgets.log_widget.HTMLFormatter(fmt=None, datefmt=None, style='%',
                                                                     validate=True, *, defaults=None)
     Bases: Formatter
     format(record)
           Format the specified record as text.
           The record's attribute dictionary is used as the operand to a string formatting operation which yields the
           returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message
           attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time
           (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception
           information, it is formatted using formatException() and appended to the message.
class pymeasure.display.widgets.log_widget.LogWidget(name, parent=None, fmt=None,
                                                                datefmt=None)
     Bases: TabWidget, QWidget
     Widget to display logging information in GUI
     It is recommended to include this widget in all subclasses of ManagedWindowBase
class pymeasure.display.widgets.plot_frame.PlotFrame(x_axis=None, y_axis=None, refresh_time=0.2,
                                                                check_status=True, parent=None)
     Bases: QFrame
     Combines a PyQtGraph Plot with Crosshairs. Refreshes the plot based on the refresh time, and allows the axes
     to be changed on the fly, which updates the plotted data
     ResultsClass
           alias of ResultsCurve
     parse_axis(axis)
           Returns the units of an axis by searching the string
class pymeasure.display.widgets.plot_widget.PlotWidget(name, columns, x_axis=None, y_axis=None,
                                                                   refresh_time=0.2, check_status=True,
                                                                   linewidth=1, parent=None)
```

```
Bases: TabWidget, QWidget
     Extends PlotFrame to allow different columns of the data to be dynamically chosen
     clear_widget()
          Clear widget content
          Behaviour is widget specific and it is currently used in preview mode
     load(curve)
          Add curve to widget
     new_curve(results, color=<PyQt5.QtGui.QColor object>, **kwargs)
          Create a new curve
     preview_widget(parent=None)
          Return a widget suitable for preview during loading
     remove(curve)
          Remove curve from widget
     set_color(curve, color)
          Change the color of the pen of the curve
     sizeHint(self) \rightarrow QSize
class pymeasure.display.widgets.results_dialog.ResultsDialog(procedure_class, widget_list=(),
                                                                         parent=None)
     Bases: QFileDialog
     Widget that displays a dialog box for loading a past experiment run. It shows a preview of curves from the results
     file when selected in the dialog box.
     This widget used by the open_experiment method in ManagedWindowBase class
class pymeasure.display.widgets.sequencer_widget.ComboBoxDelegate(owner, choices)
     Bases: QStyledItemDelegate
     {\tt createEditor}(self, parent: QWidget \mid None, option: QStyleOptionViewItem, index: QModelIndex) 
ightarrow
                     QWidget | None
     setEditorData(self, editor: QWidget | None, index: QModelIndex)
     setModelData(self, editor: QWidget | None, model: QAbstractItemModel | None, index: QModelIndex)
     updateEditorGeometry(self, editor: QWidget | None, option: QStyleOptionViewItem, index:
                               QModelIndex)
class pymeasure.display.widgets.sequencer_widget.ExpressionValidator
     Bases: QValidator
     validate(self, a0: str | None, a1: int)
class pymeasure.display.widgets.sequencer_widget.LineEditDelegate
     Bases: QStyledItemDelegate
     createEditor(self, parent: QWidget | None, option: QStyleOptionViewItem, index: QModelIndex) \rightarrow
                     QWidget | None
```

class pymeasure.display.widgets.sequencer\_widget.SequenceDialog(save=False, parent=None)

Bases: QFileDialog

Widget that displays a dialog box for loading or saving a sequence tree.

It also shows a preview of sequence tree in the dialog box

#### **Parameters**

**save** – True if we are saving a file. Default False.

Bases: QAbstractItemModel

Model for sequencer data

#### **Parameters**

- header List of string representing header data
- data data associated with the model
- parent A QWidget that QT will give ownership of this Widget to.

# add\_node(parameter, parent=None)

Add a row in the sequencer

## columnCount(parent)

Return the number of columns in the model header.

The parent parameter exists only to support the signature of QAbstractItemModel.

# data(index, role)

Return the data to display for the given index and the given role.

This method should not be called directly. This method is called implicitly by the QTreeView that is displaying us, as the way of finding out what to display where.

## **flags**(*index*)

Set the flags for the item at the given QModelIndex.

Here, we just set all indexes to enabled, and selectable.

## headerData(section, orientation, role)

Return the header data for the given section, orientation and role.

This method should not be called directly. This method is called implicitly by the QTreeView that is displaying us, as the way of finding out what to display where.

# index(row, col, parent)

Return a QModelIndex instance pointing the row and column underneath the parent given. This method should not be called directly. This method is called implicitly by the QTreeView that is displaying us, as the way of finding out what to display where.

```
parent(index=None)
```

Return the index of the parent of a given index. If index is not supplied, return an invalid QModelIndex.

## **Parameters**

**index** – QModelIndex optional.

Returns

## remove\_node(index)

Remove a row in the sequencer

#### rowCount(parent)

Return the number of children of a given parent.

If an invalid QModelIndex is supplied, return the number of children under the root.

#### **Parameters**

parent - QModelIndex

 $\textbf{setData}(\textit{self}, \textit{index}: \textit{QModelIndex}, \textit{value}: \textit{Any}, \textit{role}: \textit{int} = \textit{Qt.ItemDataRole}. \textit{EditRole}) \rightarrow \textbf{bool}$ 

visit\_tree(parent)

Return a generator to enumerate all the nodes in the tree

class pymeasure.display.widgets.sequencer\_widget.SequencerTreeView(parent=None)

Bases: OTreeView

**setModel**(self, model: QAbstractItemModel | None)

class pymeasure.display.widgets.sequencer\_widget.SequencerWidget(inputs=None,

sequence\_file=None,

parent=None)

Bases: QWidget

Widget that allows to generate a sequence of measurements

It allows sweeping parameters and moreover, one can write a simple text file to easily load a sequence. Sequences can also be saved

Currently requires a queue function of the *ManagedWindow* to have a "procedure" argument.

## **Parameters**

inputs – List of strings representing the parameters name

load\_sequence(\*, filename=None)

Load a sequence from a .txt file.

#### **Parameters**

**filename** – Filename (string) of the to-be-loaded file.

## queue\_sequence()

Obtain a list of parameters from the sequence tree, enter these into procedures, and queue these procedures.

class pymeasure.display.widgets.tab\_widget.TabWidget(name, \*args, \*\*kwargs)

Bases: object

Utility class to define default implementation for some basic methods.

When defining a widget to be used in subclasses of ManagedWindowBase, users should inherit from this class and provide an implementation of these methods

# clear\_widget()

Clear widget content

Behaviour is widget specific and it is currently used in preview mode

#### load(curve)

Add curve to widget

new\_curve(\*args, \*\*kwargs)

Create a new curve

#### preview\_widget(parent=None)

Return a Qt widget suitable for preview during loading

See also ResultsDialog If the object returned is not None, then it should have also an attribute name.

## remove(curve)

Remove curve from widget

set\_color(curve, color)

Set color for widget

class pymeasure.display.widgets.dock\_widget.DockWidget(name, procedure\_class,

x\_axis\_labels=None, y\_axis\_labels=None, linewidth=1, layout\_path='./', layout\_filename='', parent=None)

Bases: TabWidget, QWidget

Widget that contains a DockArea with a number of Docks as determined by the length of the longest  $x_axis_labels$  or  $y_axis_labels$  list.

### **Parameters**

- name Name for the TabWidget
- **procedure\_class** procedure class describing the experiment (see *Procedure*)
- x\_axis\_labels List of data column(s) for the x-axis of the plot. If the list is shorter than y\_axis\_labels the last item in the list to match y\_axis\_labels length.
- y\_axis\_labels List of data column(s) for the y-axis of the plot. If the list is shorter than x\_axis\_labels the last item in the list to match x\_axis\_labels length.
- linewidth line width for plots in *PlotWidget*
- layout\_path Directory path to save dock layout state. Default is './'
- layout\_filename Optional filename for dock layout file. Default: *current procedure class* + "\_dock\_layout.json"
- parent Passed on to QtWidgets.QWidget. Default is None

contextMenuEvent(self, a0: QContextMenuEvent | None)

new\_curve(results, color=<PyQt5.QtGui.QColor object>, \*\*kwargs)

Create a new curve

#### save\_dock\_layout()

Save the current layout of the docks and the plot settings. When running the GUI you can access this function by right-clicking in the widget area to bring up the context menu and selecting "Save Dock Layout"

Bases: QAbstractTableModel

This class provided a model to manage multiple panda dataframes and display them as a single table.

The multiple pandas dataframes are provided as ResultTable class instances and all of them share the same number of columns.

There are some assumptions: - Series in the dataframe are identical, we call this number k - Series length can be different, we call this number l(x), where x=1..n

The data can be presented as follow: - By column: each series in a separate column, in this case table shape will be:  $(k*n) \times (\max(l(x) \times 1..n))$  - By row: column fixed to the number of series, in this case table shape will be:  $k \times (\sup of l(x) \times 1..n)$ 

 $columnCount(self, parent: QModelIndex = QModelIndex()) \rightarrow int$ 

 $data(self, index: QModelIndex, role: int = Qt.ItemDataRole.DisplayRole) \rightarrow Any$ 

headerData(section, orientation, role)

Return header information

Override method from QAbstractTableModel

## pandas\_column\_count()

Return total column count of the panda dataframes

The value depends on the geometry selected to display dataframes

#### pandas\_row\_count()

Return total row count of the panda dataframes

The value depends on the geometry selected to display dataframes

 $rowCount(self, parent: QModelIndex = QModelIndex()) \rightarrow int$ 

## translate\_to\_global(results, row, col)

Translate from single results coordinates to full table coordinates

#### translate\_to\_local(row, col)

Translate from full table coordinate to single results coordinates

Bases: PandasModelBase

## pandas\_column\_count()

Return total column count of the panda dataframes

The value depends on the geometry selected to display dataframes

## pandas\_row\_count()

Return total row count of the panda dataframes

The value depends on the geometry selected to display dataframes

## translate\_to\_global(results, row, col)

Translate from single results coordinates to full table coordinates

```
translate_to_local(row, col)
          Translate from full table coordinate to single results coordinates
class pymeasure.display.widgets.table_widget.PandasModelByRow(column_index=None,
                                                                          results list=[], parent=None)
     Bases: PandasModelBase
     pandas_column_count()
          Return total column count of the panda dataframes
          The value depends on the geometry selected to display dataframes
     pandas_row_count()
          Return total row count of the panda dataframes
          The value depends on the geometry selected to display dataframes
     translate_to_global(results, row, col)
          Translate from single results coordinates to full table coordinates
     translate_to_local(row, col)
          Translate from full table coordinate to single results coordinates
class pymeasure.display.widgets.table_widget.ResultsTable(results, color, column_index=None,
                                                                     force_reload=False, wdg=None,
                                                                     **kwargs)
     Bases: QObject
     Class representing a panda dataframe
class pymeasure.display.widgets.table_widget.Table(refresh_time=0.2, check_status=True,
                                                            force_reload=False, layout_class=<class 'pymea-
                                                            sure.display.widgets.table_widget.PandasModelByColumn'>,
                                                             column_index=None, float_digits=6,
                                                             parent=None)
     Bases: QTableView
     Table format view of Experiment objects
     setModel(self, model: QAbstractItemModel | None)
     set_model(model_class)
          Replace model with new instance of model class
class pymeasure.display.widgets.table_widget.TableWidget(name, columns, by_column=True,
                                                                    column_index=None, refresh_time=0.2,
                                                                   float_digits=6, check_status=True,
                                                                   parent=None)
     Bases: TabWidget, QWidget
     Widget to display experiment data in a tabular format
     clear_widget()
          Clear widget content
          Behaviour is widget specific and it is currently used in preview mode
     load(table)
          Add curve to widget
```

# 6.12 Windows classes

Bases: ManagedWindow

Display experiment output with an ImageWidget class.

#### **Parameters**

- procedure\_class procedure class describing the experiment (see *Procedure*)
- **x\_axis** the data-column for the x-axis of the plot, cannot be changed afterwards for the image-plot
- **y\_axis** the data-column for the y-axis of the plot, cannot be changed afterwards for the image-plot
- z\_axis the initial data-column for the z-axis of the plot, can be changed afterwards
- \*\*kwargs optional keyword arguments that will be passed to ManagedWindow

Bases: ManagedWindowBase

Display experiment output with an PlotWidget class.



# Tutorial Using the ManagedWindow

A tutorial and example on the basic configuration and usage of ManagedWindow.

# **Parameters**

- **procedure\_class** procedure class describing the experiment (see *Procedure*)
- **x\_axis** the initial data-column for the x-axis of the plot
- **y\_axis** the initial data-column for the y-axis of the plot

6.12. Windows classes 91

- linewidth linewidth for the displayed curves, default is 1
- **log\_fmt** formatting string for the log-widget
- **log\_datefmt** formatting string for the date in the log-widget
- \*\*kwargs optional keyword arguments that will be passed to ManagedWindowBase

class pymeasure.display.windows.managed\_window.ManagedWindowBase(procedure\_class,

widget list=(), inputs=(),displays=(), log\_channel=", log\_level=20, parent=None, sequencer=False, sequencer inputs=None, sequence\_file=None, inputs\_in\_scrollarea=False, enable\_file\_input=True, hide\_groups=True)

Bases: QMainWindow

Base class for GUI experiment management.

The ManagedWindowBase provides an interface for inputting experiment parameters, running several experiments (*Procedure*), plotting result curves, and listing the experiments conducted during a session.

The ManagedWindowBase uses a Manager to control Workers in a Queue, and provides a simple interface. The queue() method must be overridden by the child class.

The ManagedWindowBase allow user to define a set of widget that display information about the experiment. The information displayed may include: plots, tabular view, logging information,...

This class is not intended to be used directly, but it should be subclassed to provide some appropriate widget list. Example of classes usable as element of widget list are:

- LogWidget
- PlotWidget
- ImageWidget

Of course, users can define its own widget making sure that inherits from TabWidget.

Examples of ready to use classes inherited from ManagedWindowBase are:

- ManagedWindow
- ManagedImageWindow



# See also

## **Tutorial** *Using the ManagedWindow*

A tutorial and example on the basic configuration and usage of ManagedWindow.

Parameters for \_\_init\_\_ constructor.

## **Parameters**

- **procedure\_class** procedure class describing the experiment (see *Procedure*)
- widget\_list list of widget to be displayed in the GUI

- inputs list of Parameter instance variable names, which the display will generate graphical fields for
- displays list of Parameter instance variable names displayed in the browser window
- log\_channel logging.Logger instance to use for logging output
- log\_level logging level
- parent Parent widget or None
- sequencer a boolean stating whether or not the sequencer has to be included into the window
- **sequencer\_inputs** either None or a list of the parameter names to be scanned over. If no list of parameters is given, the parameters displayed in the manager queue are used.
- **sequence\_file** simple text file to quickly load a pre-defined sequence with the Load sequence button
- inputs\_in\_scrollarea boolean that display or hide a scrollbar to the input area
- enable\_file\_input a boolean controlling whether a FileInputWidget to specify where the experiment's result will be saved is displayed (True, default) or not (False). This widget contains a field to enter the (base of the) filename (with or without extension; if absent, the extension will be appended). This field also allows for placeholders to use parameter-values and metadata-value in the filename. The widget also has a field to select the directory where the file is to be stored, and a toggle to control whether the data should be saved to the selected file, or not (i.e., to a temporary file instead).
- **hide\_groups** a boolean controlling whether parameter groups are hidden (True, default) or disabled/grayed-out (False) when the group conditions are not met.

# open\_file\_externally(filename)

Method to open the datafile using an external editor or viewer. Uses the default application to open a datafile of this filetype, but can be overridden by the child class in order to open the file in another application of choice.

#### **queue**(procedure=None)

Queue a measurement based on the parameters in the input-widget.

Semi-abstract method, which must be overridden by the child class if the filename- and directory-inputs are disabled. When filename- and directory inputs are enabled, overwriting is not required, but can be done for custom naming, input processing, or other features.

Implementations must call self.manager.queue(experiment) and pass an experiment (*Experiment*) object which contains the *Results* and *Procedure* to be run.

The optional *procedure* argument is not required for a basic implementation, but is required when the *SequencerWidget* is used.

For example:

6.12. Windows classes 93

(continued from previous page)

self.manager.queue(experiment)

### reveal\_in\_file\_explorer(filename: str) $\rightarrow$ None

Method to open the file explorer at the location of the given filename.

### Args:

filename (str): Path to the file to be revealed in the file explorer.

#### save\_experiment\_copy(source\_filename)

Save a copy of the datafile to a selected folder and file. Primarily useful for experiments that are stored in a temporary file.

#### set\_parameters(parameters)

This method should be overwritten by the child class. The parameters argument is a dictionary of Parameter objects. The Parameters should overwrite the GUI values so that a user can click "Queue" to capture the same parameters.

Bases: QMainWindow

A window for plotting experiment results. Should not be instantiated directly, but only via the *Plotter* class.

## See also

Tutorial *Using the Plotter* A tutorial and example code for using the Plotter and PlotterWindow.

## check\_stop()

Checks if the Plotter should stop and exits the Qt main loop if so

class pymeasure.display.windows.managed\_dock\_window.ManagedDockWindow(procedure\_class,

x\_axis=None, y\_axis=None, linewidth=1, log\_fmt=None, log\_datefmt=None, \*\*kwargs)

Bases: ManagedWindowBase

Display experiment output with multiple docking windows with *DockWidget* class.

#### **Parameters**

- **procedure\_class** procedure class describing the experiment (see *Procedure*)
- **x\_axis** the data column(s) for the x-axis of the plot. This may be a string or a list of strings from the data columns of the procedure. The list length determines the number of plots
- **y\_axis** the data column(s) for the y-axis of the plot. This may be a string or a list of strings from the data columns of the procedure. The list length determines the number of plots
- linewidth linewidth for the displayed curves, default is 1
- log\_fmt formatting string for the log-widget
- log\_datefmt formatting string for the date in the log-widget

• \*\*kwargs – optional keyword arguments that will be passed to ManagedWindowBase

6.12. Windows classes 95

PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20250827				

# **PYMEASURE.INSTRUMENTS**

This section contains documentation on the instrument classes.

# 7.1 Instrument classes

class pymeasure.instruments.common\_base.CommonBase(\*\*kwargs)

Base class for instruments and channels.

This class contains everything needed for pymeasure's property creator *control()* and its derivatives *measurement()* and *setting()*.

```
class BaseChannelCreator(cls, **kwargs)
```

Base class for ChannelCreator and MultiChannelCreator.

## **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- **\*\*kwargs** Keyword arguments for all children.

```
class ChannelCreator(cls, id=None, **kwargs)
```

Add a single channel to the parent class.

The child will be added to the parent instance at instantiation with <code>CommonBase.add\_child()</code>. The attribute name that ChannelCreator was assigned to in the <code>Instrument</code> class will be the name of the channel interface.

```
class Extreme5000(Instrument):
    # Two output channels, accessible by their property names
    # and both are accessible through the 'channels' collection
    output_A = Instrument.ChannelCreator(Extreme5000Channel, "A")
    output_B = Instrument.ChannelCreator(Extreme5000Channel, "B")
    # A channel without a channel accessible through the 'motor' collection
    motor = Instrument.ChannelCreator(MotorControl)

inst = SomeInstrument()
# Set the extreme_temp for channel A of Extreme5000 instrument
inst.output_A.extreme_temp = 42
```

### **Parameters**

- cls Channel class for channel interface
- id The id of the channel on the instrument, integer or string.

• **\*\*kwargs** – Keyword arguments for all children.

**class MultiChannelCreator**(cls, id=None, prefix='ch', \*\*kwargs)

Add channels to the parent class.

The children will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name (e.g. channels) will be used as the collection of the children. You may define the attribute prefix. If there are no other pressing reasons, use channels as the attribute name and leave the prefix at the default "ch ".

```
class Extreme5000(Instrument):
    # Three channels of the same type: 'ch_A', 'ch_B', 'ch_C'
    # and add them to the 'channels' collection
    channels = Instrument.MultiChannelCreator(Extreme5000Channel, ["A", "B", "C
''])
    # Two channel interfaces of different types: 'fn_power', 'fn_voltage'
    # and add them to the 'functions' collection
    functions = Instrument.MultiChannelCreator((PowerChannel, VoltageChannel),
                                     ["power", "voltage"], prefix="fn_")
```

#### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- id tuple/list of ids of the channels on the instrument.
- **prefix** Collection prefix for the attributes, e.g. "ch\_" creates attribute self.ch\_A. If prefix evaluates False, the child will be added directly under the variable name. Required if id is tuple/list.
- **\*\*kwargs** Keyword arguments for all children.

add\_child(cls, id=None, collection='channels', prefix='ch\_', attr\_name='', \*\*kwargs)

Add a child to this instance and return its index in the children list.

The newly created child may be accessed either by the id in the children dictionary or by the created attribute, e.g. the fifth channel of *instrument* with id "F" has two access options: instrument.channels["F"] == instrument.ch\_F



# 1 Note

Do not change the default collection or prefix parameter, unless you have to distinguish several collections of different children, e.g. different channel types (analog and digital).

#### **Parameters**

- **cls** Class of the channel.
- id Child id how it is used in communication, e.g. "A".
- collection Name of the collection of children, used for dictionary access to the channel interfaces.
- prefix For creating multiple channel interfaces, the prefix e.g. "ch\_" is prepended to the attribute name of the channel interface self.ch\_A. If prefix evaluates False, the child will be added directly under the collection name.

- attr\_name For creating a single channel interface, the attr\_name argument is used when setting the attribute name of the channel interface.
- **\*\*kwargs** Keyword arguments for the channel creator.

#### Returns

Instance of the created child.

## ask(command, query\_delay=None)

Write a command to the instrument and return the read response.

#### **Parameters**

- **command** Command string to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.

#### Returns

String returned by the device without read\_termination.

## binary\_values(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return a numpy array of the binary data.

#### **Parameters**

- **command** Command to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- **kwargs** Arguments for read\_binary\_values().

#### Returns

NumPy array of values.

# check\_errors()

Read all errors from the instrument and log them.

#### **Returns**

List of error entries.

## check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

## Returns

List of error entries.

## check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

## Returns

List of error entries.

7.1. Instrument classes 99

static control(get\_command, set\_command, docs, validator=<function CommonBase.<lambda>>, values=(), map\_values=False, get\_process=<function CommonBase.<lambda>>, get\_process\_list=<function CommonBase.<lambda>>, set\_process=<function CommonBase.<lambda>>, command\_process=None, check\_set\_errors=False, check\_get\_errors=False, dynamic=False, preprocess\_reply=None, separator=', ', maxsplit=-1, cast=<class 'float'>, values\_kwargs=None, \*\*kwargs')

Return a property for the class based on the supplied commands. This property may be set and read from the instrument. See also *measurement()* and *setting()*.

#### **Parameters**

- **get\_command** A string command that asks for the value, set to *None* if get is not supported (see also *setting()*).
- **set\_command** A string command that writes the value, set to *None* if set is not supported (see also *measurement* ()).
- **docs** A docstring that will be included in the documentation
- validator A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that takes a command and allows processing before executing the command

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_set\_errors** Toggles checking errors after setting
- **check\_get\_errors** Toggles checking errors after getting
- dynamic Specify whether the property parameters are meant to be changed in instances or subclasses.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- values\_kwargs (dict) Further keyword arguments for values().
- \*\*kwargs Keyword arguments for *values()*.

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

Example of usage of dynamic parameter is as follows:

```
class GenericInstrument(Instrument):
    center_frequency = Instrument.control(
        ":SENS:FREQ:CENT?;", ":SENS:FREQ:CENT %e GHz;",
        " A floating point property that represents the frequency ... ",
        validator=strict_range,
        # Redefine this in subclasses to reflect actual instrument value:
        values=(1, 20),
        dynamic=True # enable changing property parameters on-the-fly
    )

class SpecificInstrument(GenericInstrument):
    # Identical to GenericInstrument, except for frequency range
    # Override the "values" parameter of the "center_frequency" property
    center_frequency_values = (1, 10) # Redefined at subclass level

instrument = SpecificInstrument()
instrument.center_frequency_values = (1, 6e9) # Redefined at instance level
```

# **⚠** Warning

Unexpected side effects when using dynamic properties

Users must pay attention when using dynamic properties, since definition of class and/or instance attributes matching specific patterns could have unwanted side effect. The attribute name pattern *property\_param*, where *property* is the name of the dynamic property (e.g. *center\_frequency* in the example) and *param* is any of this method parameters name except *dynamic* and *docs* (e.g. *values* in the example) has to be considered reserved for dynamic property control.

#### static get\_channel\_pairs(cls)

Return a list of all the Instrument's channel pairs

```
static get_channels(cls)
```

Return a list of all the Instrument's ChannelCreator and MultiChannelCreator instances

Return a property for the class based on the supplied commands. This is a measurement quantity that may only be read from the instrument, not set.

#### **Parameters**

- get\_command A string command that asks for the value
- docs A docstring that will be included in the documentation
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value

7.1. Instrument classes 101

- **get\_process\_list** A function that takes the value list and processes it.
- **command\_process** A function that take a command and allows processing before executing the command, for getting

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_get\_errors** Toggles checking errors after getting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most maxsplit times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- values\_kwargs (dict) Further keyword arguments for values().
- \*\*kwargs Keyword arguments for values().

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

# remove\_child(child)

Remove the child from the instrument and the corresponding collection.

#### Parameters

**child** – Instance of the child to delete.

Return a property for the class based on the supplied commands. This property may be set, but raises an exception when being read from the instrument.

## **Parameters**

- **set\_command** A string command that writes the value
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **check\_set\_errors** Toggles checking errors after setting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.

**values**(*command*, *separator='*, ', *cast=<class 'float'>*, *preprocess\_reply=None*, *maxsplit=-1*, \*\*kwargs)
Write a command to the instrument and return a list of formatted values from the result.

## **Parameters**

- **command** SCPI command to be sent to the instrument.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most maxsplit times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- **\*\*kwargs** Keyword arguments to be passed to the *ask()* method.

#### Returns

A list of the desired type, or strings where the casting fails.

#### wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

Implement in subclass!

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

class pymeasure.instruments.Instrument(adapter, name, includeSCPI=None, \*\*kwargs)

The base class for all Instrument definitions.

It makes use of one of the *Adapter* classes for communication with the connected hardware device. This decouples the instrument/command definition from the specific communication interface used.

When adapter is a string, this is taken as an appropriate resource name. Depending on your installed VISA library, this can be something simple like COM1 or ASRL2, or a more complicated VISA resource name defining the target of your connection.

When adapter is an integer, a GPIB resource name is created based on that. In either case a *VISAAdapter* is constructed based on that resource name. Keyword arguments can be used to further configure the connection.

Otherwise, the passed Adapter object is used and any keyword arguments are discarded.

This class defines basic SCPI commands by default. This can be disabled with includeSCPI for instruments not compatible with the standard SCPI commands.

#### **Parameters**

- adapter A string, integer, or Adapter subclass object
- name (string) The name of the instrument. Often the model designation by default.
- includeSCPI An obligatory boolean, which toggles the inclusion of standard SCPI commands

Deprecated since version 0.14: If True, inherit the SCPIMixin class instead.

• \*\*kwargs – In case adapter is a string or integer, additional arguments passed on to VISAAdapter (check there for details). Discarded otherwise.

#### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

7.1. Instrument classes 103

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# clear()

Clears the instrument status byte

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property id

Get the identification of the instrument.

# property next\_error

Get the next error of the instrument (tuple of code and message).

#### property options

Get the device options installed.

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

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

# **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

#### Returns bytes

Bytes response of the instrument (including termination).

# reset()

Resets the instrument.

#### shutdown()

Brings the instrument to a safe and stable state

# property status

Get the status byte and Master Summary Status bit.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# class pymeasure.instruments.Channel(parent, id)

The base class for channel definitions.

This class supports dynamic properties like *Instrument*, but requires an *Instrument* instance as a parent for communication.

*insert\_id()* inserts the channel id into the command string sent to the instrument. The default implementation replaces the Channel's *placeholder* (default "ch") with the channel id in all command strings (e.g. "CHANnel{ch}:foo").

#### **Parameters**

- parent The instrument (an instance of *Instrument*) to which the channel belongs.
- id Identifier of the channel, as it is used for the communication.

# check\_errors()

Read all errors from the instrument and log them.

#### **Returns**

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

7.1. Instrument classes 105

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# insert\_id(command)

Insert the channel id in a command replacing placeholder.

Subclass this method if you want to do something else, like always prepending the channel id.

# read(\*\*kwargs)

Read up to (excluding) *read\_termination* or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the instrument.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

#### wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write termination.

#### **Parameters**

- **command** command string to be sent to the instrument. '{ch}' is replaced by the channel id.
- **kwargs** Keyword arguments for the adapter.

# write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the instrument.

#### **Parameters**

- **command** Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

Bases: Instrument

Provides a fake implementation of the Instrument class for testing purposes.

Fake Instrument.control.

Strip commands and only store and return values indicated by format strings to mimic many simple commands. This is analogous how the tests in test\_instrument are handled.

class pymeasure.instruments.fakes.SwissArmyFake(name='Mock instrument', wait=0.1, \*\*kwargs)

Bases: FakeInstrument

Dummy instrument class useful for testing.

Like a Swiss Army knife, this class provides multi-tool functionality in the form of streams of multiple types of fake data. Data streams that can currently be generated by this class include 'voltages', sinusoidal 'waveforms', and mono channel 'image data'.

# property frame

Get a new image frame.

# property frame\_format

Control the format for image data returned from the get\_frame() method. Allowed values are: mono\_8: single channel 8-bit image. mono\_16: single channel 16-bit image.

# property frame\_height

Control frame height in pixels.

# property frame\_width

Control frame width in pixels.

# property output\_voltage

Control the voltage.

# property time

Control the elapsed time.

# property voltage

Measure the voltage.

# property wave

Measure a waveform.

7.1. Instrument classes 107

# 7.2 Generic Instrument Types Mixins

You can use these Mixins as additional parent classes for your instrument. For more informations, see *Common instrument types*.

#### class pymeasure.instruments.generic\_types.SCPIMixin(\*args, \*\*kwargs)

Mixin class for SCPI instruments with the default implementation of base SCPI commands.

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

#### clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

#### property id

Get the identification of the instrument.

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

#### property options

Get the device options installed.

#### reset()

Reset the instrument.

# property status

Get the status byte and Master Summary Status bit.

# 7.3 Validator functions

Validators are used in conjunction with the Instrument.control or Instrument.setting functions to allow properties with complex restrictions for valid values. They are described in more detail in the *Restricting values with validators* section.

# pymeasure.instruments.validators.discreteTruncate(number, discreteSet)

Truncates the number to the closest element in the positive discrete set. Returns False if the number is larger than the maximum value or negative.

# pymeasure.instruments.validators.joined\_validators(\*validators)

Returns a validator function that represents a list of validators joined together.

A value passed to the validator is returned if it passes any validator (not all of them). Otherwise it raises a ValueError.

Note: the joined validator expects values to be a sequence of values appropriate for the respective validators (often sequences themselves).

# **Example**

```
>>> from pymeasure.instruments.validators import strict_discrete_set, strict_range
>>> from pymeasure.instruments.validators import joined_validators
>>> joined_v = joined_validators(strict_discrete_set, strict_range)
>>> values = [['MAX','MIN'], range(10)]
>>> joined_v(5, values)
5
>>> joined_v('MAX', values)
'MAX'
>>> joined_v('NONSENSE', values)
Traceback (most recent call last):
...
ValueError: Value of NONSENSE does not match any of the joined validators
```

#### **Parameters**

**validators** – an iterable of other validators

pymeasure.instruments.validators.modular\_range(value, values)

Provides a validator function that returns the value if it is in the range. Otherwise it returns the value, modulo the max of the range.

#### **Parameters**

- value a value to test
- values A set of values that are valid

pymeasure.instruments.validators.modular\_range\_bidirectional(value, values)

Provides a validator function that returns the value if it is in the range. Otherwise it returns the value, modulo the max of the range. Allows negative values.

# **Parameters**

- value a value to test
- values A set of values that are valid

```
pymeasure.instruments.validators.strict_discrete_range(value, values, step)
```

Provides a validator function that returns the value if its value is less than the maximum and greater than the minimum of the range and is a multiple of step. Otherwise it raises a ValueError.

#### **Parameters**

- value A value to test
- **values** A range of values (range, list, etc.)
- **step** Minimum stepsize (resolution limit)

#### Raises

ValueError if the value is out of the range

```
pymeasure.instruments.validators.strict_discrete_set(value, values)
```

Provides a validator function that returns the value if it is in the discrete set. Otherwise it raises a ValueError.

#### **Parameters**

- value A value to test
- values A set of values that are valid

7.3. Validator functions 109

#### Raises

ValueError if the value is not in the set

pymeasure.instruments.validators.strict\_range(value, values)

Provides a validator function that returns the value if its value is less than or equal to the maximum and greater than or equal to the minimum of values. Otherwise it raises a ValueError.

#### **Parameters**

- value A value to test
- **values** A range of values (range, list, etc.)

# **Raises**

ValueError if the value is out of the range

pymeasure.instruments.validators.truncated\_discrete\_set(value, values)

Provides a validator function that returns the value if it is in the discrete set. Otherwise, it returns the smallest value that is larger than the value.

#### **Parameters**

- value A value to test
- values A set of values that are valid

pymeasure.instruments.validators.truncated\_range(value, values)

Provides a validator function that returns the value if it is in the range. Otherwise it returns the closest range bound.

#### **Parameters**

- value A value to test
- values A set of values that are valid

# 7.4 Comedi data acquisition

The Comedi libraries provide a convenient method for interacting with data acquisition cards, but are restricted to Linux compatible operating systems.

```
\verb|pymeasure.instruments.comedi.getAI| (device, channel, range=None)|
```

Returns the analog input channel as specified for a given device

```
pymeasure.instruments.comedi.getAO(device, channel, range=None)
```

Returns the analog output channel as specified for a given device

```
pymeasure.instruments.comedi.readAI(device, channel, range=None, count=1)
```

Reads a single measurement (count==1) from the analog input channel of the device specified. Multiple readings can be preformed with count not equal to one, which are separated by an arbitrary time

```
pymeasure.instruments.comedi.writeAO(device, channel, voltage, range=None)
```

Writes a single voltage to the analog output channel of the device specified

# 7.5 Resource Manager

The list\_resources function provides an interface to check connected instruments interactively.

```
pymeasure.instruments.list_resources()
```

Prints the available resources, and returns a list of VISA resource names

```
resources = list_resources()
#prints (e.g.)
    #0 : GPIB0::22::INSTR : Agilent Technologies,34410A,*****
    #1 : GPIB0::26::INSTR : Keithley Instruments Inc., Model 2612, *****
dmm = Agilent34410(resources[0])
```

Instruments by manufacturer:

# 7.6 Active Technologies

This section contains specific documentation on the Active Technologies instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.6.1 Active Technologies AWG-401x 1.2GS/s Arbitrary Waveform Generator

class pymeasure.instruments.activetechnologies.AWG401x\_AFG(adapter, \*\*kwargs)
 Bases: AWG401x base

Represents the Active Technologies AWG-401x Arbitrary Waveform Generator in AFG mode.

```
wfg = AWG401x_AFG("TCPIP::192.168.0.123::INSTR")
wfg.reset()
                                 # Reset the instrument at default state
wfg.channels[1].shape = "SINUSOID"
                                      # Sets a sine waveform on CH1
wfg.channels[1].frequency = 4.7e3  # Sets the frequency to 4.7 kHz on CH1
wfg.channels[1].amplitude = 1  # Set amplitude of 1 V on CH1
wfg.channels[1].offset = 0
                                     # Set the amplitude to 0 V on CH1
wfg.channels[1].enabled = True
                                      # Enables the CH1
wfq.channels[2].shape = "SQUARE" # Sets a square waveform on CH2
wfg.channels[2].frequency = 100e6  # Sets the frequency to 100 MHz on CH2
wfg.channels[2].amplitude = 0.5  # Set amplitude of 0.5 V on CH2
wfg.channels[2].offset = 0
                                    # Set the amplitude to 0 V on CH2
wfg.channels[2].enabled = True
                                      # Enables the CH2
wfg.enabled = True
                                 # Enable output of waveform generator
                                 # "beep"
wfg.beep()
print(wfg.check_errors())
                                 # Get the error queue
```

ch 1

Channel

Channel AFG

#### $ch_2$

#### Channel

Channel AFG

#### property enabled

Control whether the generation of signals is enabled (bool).

class pymeasure.instruments.activetechnologies.AWG401x\_AWG(adapter, \*\*kwargs)

Bases: AWG401x base

Represents the Active Technologies AWG-401x Arbitrary Waveform Generator in AWG mode.

```
wfg = AWG401x\_AWG("TCPIP::192.168.0.123::INSTR")
wfg.reset()
                       # Reset the instrument at default state
# Set a oscillating waveform
wfg.waveforms["MyWaveform"] = [1, 0] * 8
for i in range(1, wfg.num_ch + 1):
   wfg.entries[1].channels[i].voltage_high = 1  # Sets high voltage = 1
wfg.entries[1].channels[i].voltage_low = 0  # Sets low voltage = 1
   wfg.setting_ch[i].enabled = True
                                              # Enable channel
wfg.entries.resize(2)
                               # Resize the number of entries to 2
wfg.entries[2].channels[1].waveform = "MyWaveform"
                                                    # Set custom waveform
wfg.enabled = True
                               # Enable output of waveform generator
wfg.beep()
                               # "beep"
print(wfg.check_errors())
                               # Get the error queue
```

#### class DummyEntriesElements(parent, number of channel)

Bases: Sequence

Dummy List Class to list every sequencer entry. The content is loaded in real-time.

# class WaveformsLazyDict(parent)

Bases: MutableMapping

This class inherit from MutableMapping in order to create a custom dict to lazy load, modify, delete and create instrument waveform.

#### reset()

Reset the class reloading the waveforms from instrument

# property burst\_count

Control the burst count parameter.(dynamic)

#### property burst\_count\_max

Get the maximum burst count parameter.

# property burst\_count\_min

Get the minimum burst count parameter.

#### property enabled

Control whether generation of signals in enabled.

# property entry\_level\_strategy

Control the Entry Length Strategy. This strategy manages the length of the sequencer entries in relationship with the length of the channel waveforms defined for each entry. The possible values are:

- ADAPTL<ONGER>: the length of an entry of the sequencer by default will be equal to the length of the longer channel waveform, among all analog channels, assigned to the entry.
- ADAPTS<HORTER>: the length of an entry of the sequencer by default will be equal to the length of the shorter channel waveform, among all analog channels, assigned to the entry.
- DEF<AULT>:the length of an entry of the sequencer by default will be equal to the value specified in the Sequencer Item Default Length [N] parameter

# list\_files(path=None)

Return a List of tuples with all file found in a directory. If the path is not specified the current directory will be used

# property num\_ch

Get the number of analog channels.

# property num\_dch

Get the number of digital channels.

# remove\_file(file\_name, path=None)

Remove a specified file

# property run\_mode

Control the AWG run mode. The possible values are:

- CONT<INUOUS>: each waveform will loop as written in the entry repetition parameter and the entire sequence is repeated circularly
- BURS<T>: the AWG waits for a trigger event. When the trigger event occurs each waveform will loop as written in the entry repetition parameter and the entire sequence will be repeated circularly many times as written in the Burst Count[N] parameter. If you set Burst Count[N]=1 the instrument is in Single mode and the sequence will be repeated only once.
- TCON<TINUOUS>: the AWG waits for a trigger event. When the trigger event occurs each waveform will loop as written in the entry repetition parameter and the entire sequence will be repeated circularly.
- STEP<PED>: the AWG, for each entry, waits for a trigger event before the execution of the sequencer entry. The waveform of the entry will loop as written in the entry repetition parameter. After the generation of an entry has completed, the last sample of the current entry or the first sample of the next entry is held until the next trigger is received. At the end of the entire sequence the execution will restart from the first entry.
- ADVA<NCED>: it enables the "Advanced" mode. In this mode the execution of the sequence can be changed by using conditional and unconditional jumps (JUMPTO and GOTO commands) and dynamic jumps (PATTERN JUMP commands).

The \*RST command sets this parameter to CONTinuous.

#### property run\_status

Get the run state of the AWG. The possible values are: STOPPED, WAITING\_TRIGGER, RUNNING

#### property sample\_decreasing\_strategy

Control the Sample Decreasing Strategy. The "Sample decreasing strategy" parameter defines the strategy used to adapt the waveform length to the sequencer entry length in the case where the original waveform length is longer than the sequencer entry length. Can be set to: DECIM<ATION>, CUTT<AIL>, CUTH<EAD>

#### property sample\_increasing\_strategy

Control the Sample Increasing Strategy. The "Sample increasing strategy" parameter defines the strategy used to adapt the waveform length to the sequencer entry length in the case where the original waveform length is shorter than the sequencer entry length. Can be set to: INTER<POLATION>, RETURN<ZERO>, HOLD<LAST>, SAMPLESM<ULTIPLICATION>

# property sampling\_rate

Control the sample rate for the Sampling Clock.(dynamic)

#### property sampling\_rate\_max

Get the maximum sample rate for the Sampling Clock.

# property sampling\_rate\_min

Get the minimum sample rate for the Sampling Clock.

save\_file(file\_name, data, path=None, override\_existing=False)

Write a string in a file in the instrument

# trigger()

Force a trigger event to occour.

#### property trigger\_source

Control the instrument trigger source. The possible values are:

- TIM<ER>: the trigger is sent at regular intervals.
- EXT<ERNAL>: the trigger come from the external BNC connector.
- MAN<UAL>: the trigger is sent via software or using the trigger button on front panel.

# property waveforms

Get a dict with all the waveform present in the instrument system (Wave. List). It is possible to modify the values, delete them or create new waveforms

# class pymeasure.instruments.activetechnologies.AWG401x.ChannelAFG(instrument, id)

Bases: ChannelBase

Implementation of a Active Technologies AWG-4000 channel in AFG mode.

#### property baseline\_offset

Control the offset level for the specified channel. The offset range setting depends on the amplitude parameter. (dynamic)

# property baseline\_offset\_max

Get the maximum offset voltage level that can be set to the output waveform.

#### property baseline\_offset\_min

Get the minimum offset voltage level that can be set to the output waveform.

# property frequency

Control the frequency of the output waveform. This command is available when the Run Mode is set to any setting other than Sweep. The output frequency range setting depends on the type of output waveform. If you change the type of output waveform, it may change the output frequency because changing waveform

types affects the setting range of the output frequency. The output frequency range setting depends also on the amplitude parameter.(dynamic)

# property frequency\_max

Get the maximum frequency that can be set to the output waveform.

# property frequency\_min

Get the minimum frequency that can be set to the output waveform.

#### property load\_impedance

Control the output load impedance for the specified channel. The specified value is used for amplitude, offset, and high/low level settings. You can set the impedance to any value from 1 to 1 M. The default value is 50.

#### property output\_impedance

Control the instrument output impedance, the possible values are: 5 Ohm or 50 Ohm (default).

# property phase

Control the phase of the output waveform for the specified channel. The value is in degrees.(dynamic)

#### property phase\_max

Get the maximum phase that can be set to the output waveform.

#### property phase\_min

Get the minimum phase that can be set to the output waveform.

#### property shape

Control the shape of the carrier waveform. Allowed choices depends on the choosen modality, please refer on instrument manual. When you set this property with a different value, if the instrument is running it will be stopped. Can be set to: SIN<USOID>, SQU<ARE>, PULS<E>, RAMP, PRN<OISE>, DC, SINC, GAUS<SIAN>, LOR<ENTZ>, ERIS<E>, EDEC<AY>, HAV<ERSINE>, ARBB, EFIL<E>, DOUBLEPUL<SE>

#### property voltage\_amplitude

Control the output amplitude for the specified channel. The measurement unit of amplitude depends on the selection operated using the voltage\_unit property. If the carrier is Noise the amplitude is Vpk instead of Vpp. If the carrier is DC level this command causes an error. The range of the amplitude setting could be limited by the frequency and offset parameter of the carrier waveform. (dynamic)

#### property voltage\_amplitude\_max

Get the maximum amplitude voltage level that can be set to the output waveform.

#### property voltage\_amplitude\_min

Get the minimum amplitude voltage level that can be set to the output waveform.

# property voltage\_high

Control the high level of the waveform. The high level could be limited by noise level to not exceed the maximum amplitude. If the carrier is Noise or DC level, this command and this query cause an error.(dynamic)

# property voltage\_high\_max

Get the maximum high voltage level that can be set to the output waveform.

# property voltage\_high\_min

Get the minimum high voltage level that can be set to the output waveform.

# property voltage\_low

Control the low level of the waveform. The low level could be limited by noise level to not exceed the maximum amplitude. If the carrier is Noise or DC level, this command and this query cause an error.(dynamic)

#### property voltage\_low\_max

Get the maximum low voltage level that can be set to the output waveform.

# property voltage\_low\_min

Get the minimum low voltage level that can be set to the output waveform.

# property voltage\_offset

Control the offset level for the specified channel. The offset range setting depends on the amplitude parameter. (dynamic)

#### property voltage\_offset\_max

Get the maximum offset voltage level that can be set to the output waveform.

# property voltage\_offset\_min

Get the minimum offset voltage level that can be set to the output waveform.

# property voltage\_unit

Control the units of output amplitude, the possible choices are: VPP, VRMS, DBM. This command does not affect the offset, high level, or low level of output.

# 7.7 Aculight

This section contains specific documentation on the (Lockheed Martin) Aculight instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.7.1 Aculight Argos 2400 OPO system

Bases: Instrument

Communication model for the Aculight Argos 2400 OPO system.

Note that the device can only return all values at once, which takes around 100 ms. If you need more than one value, read the *state* property instead of individual properties. Reading any property reads all properties and returns the corresponding value.

Note also, that the block temperature cannot be set remotely, only via the front panel.

Cable type: RS232 null modem female DB9 connectors: The pins 2 and 3 have to be crossed between both sides. Pin2 is RX and Pin3 is Tx, Pin 5 is ground.

#### **Parameters**

- adapter resource name
- query\_delay Delay between write and read in seconds.

#### check\_set\_errors()

Read after setting.

# property etalon

Control the etalon angle in degrees.

#### property seed\_voltage

Control the seed source tuning voltage.

#### property state: State

Get the current state of the system as a namped tuple

# Return named tuple

with 'crystal\_temperature', 'crystal\_temperature\_setpoint', 'etalon\_angle', and 'seed\_voltage'.

# property temperature

Get the current crystal temperature in °C.

# property temperature\_setpoint

Control the crystal temperature setpoint in °C.

# property version

Get the firmware version.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

# 7.8 Advantest

This section contains specific documentation on the Advantest instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.8.1 Advantest R3767CG Vector Network Analyzer

**class** pymeasure.instruments.advantest.advantestR3767CG.**AdvantestR3767CG**(adapter,

name='Advantest R3767CG', \*\*kwargs)

Bases: SCPIUnknownMixin, Instrument

Represents the Advantest R3767CG VNA. Implements controls to change the analysis range and to retrieve the data for the trace.

# property center\_frequency

Control the center Frequency in Hz.

#### property id

Get the instrument identification.

# property span\_frequency

Control the frequency span in Hz.

#### property start\_frequency

Control the starting frequency in Hz.

# property stop\_frequency

Control the stopping frequency in Hz.

# property trace\_1

Get the data array from trace 1 after formatting.

# 7.8.2 Advantest R6245/R6246 DC Voltage/Current Sources/Monitors

# **Main Classes**

Bases: AdvantestR624X

Main instrument class for Advantest R6245 DC Voltage/Current Source/Monitor

ch\_A

#### Channel

**SMUChannel** 

ch B

#### Channel

SMUChannel

Bases: AdvantestR624X

Main instrument class for Advantest R6246 DC Voltage/Current Source/Monitor

ch\_A

# Channel

**SMUChannel** 

ch\_B

#### Channel

**SMUChannel** 

Bases: SCPIUnknownMixin, Instrument

Represents the Advantest R624X series (channel A and B) SourceMeter and provides a high-level interface for interacting with the instrument.

This is the base class for both AdvantestR6245 and AdvantestR6246 devices. It's not necessary to instantiate this class directly instead create an instance of the AdvantestR6245 or AdvantestR6246 class as shown in the following example:

(continues on next page)

(continued from previous page)

```
smu.ch_A.enable_source()
                                                                   # Enables the
→source output
smu.ch_A.measure_voltage()
smu.ch_A.current_change_source = 5e-3
                                                                   # Change to 5mA
print(smu.read_measurement())
                                                                   # Read and print_

→ the voltage

smu.ch_A.standby()
                                                                   # Put channel A in ...
→ standby
```

# write(command, \*\*kwargs)

Write a string command to the instrument appending write termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

# enable\_source()

Put channel A & B into the operating state (CN).



#### 1 Note

When the 'interlock control' of the 'SCT' command is '2' and the clock signal is 'HI', it will not enter the operating state.

# standby()

Put channel A & B in standby mode (CL).

# clear\_status\_register()

Clears the Standard Event Status Register (SESR) and related queues (excluding output queues) (\*CLS).

#### property srq\_enabled

Set a boolean that controls whether the GPIB SRQ feature is enabled, takes values of True or False (S0/S1).

# **Type**

bool

The SRQ feature of the GPIB bus provides hardware handshaking between the GPIB controller card in the PC and the instrument. This allows synchronization between moving data to the PC with the state of the instrument without the need to use time delay functions.

# trigger()

Outputs the trigger signal or the start of sweep and search measurement to both A and B channels and the trigger link (XE).

Mote

- When both A channel and B channel are waiting for a trigger, both channels are triggered.
- When either channel A or B is waiting for a trigger, only the channel that is waiting for a trigger is triggered.
- When both A channel and B channel are waiting for sweep start, this will apply sweep start to
- When either channel A or B is in the sweep start waiting state, only the channel in the sweep start waiting state is started.
- · When either channel A or B is waiting for a trigger and the other is waiting for a sweep start, trigger and sweep start are applied, respectively.
- When the trigger link is ON and this is the master unit, set the \*TRG signal on the trigger link bus to TRUE.
- When the trigger link is ON and the master unit, the trigger link is activated.

#### stop()

Stops the sweep when the sweep is started by the XE command or the trigger input signal (SP).

# set\_digital\_output(values)

Outputs a 16-bit signal from the DIGITAL OUT output terminal on the rear panel. You can set up to 9 output data (DIOS). If there are multiple values specified, the data is output at intervals of about 2ms and fixed as the final data.

#### **Parameters**

values (int or list) - Digital out bit values



# 1 Note

The output of digital data to the DIGITAL OUT pin is only the bits specified by the DIOE command. Bits that are not specified will result in alarm output or unused, and no digital data will be output.

# property sweep\_delay\_time

Set the sweep delay time (Ta) or generation / delay time (Ta) of the master channel and slave channel during delayed sweep operation or synchronous operation between pulse measurements (GDLY).

# **Type**

float



#### 1 Note

If the sweep delay time does not meet (Ta<Tw and Ta<Td+Tit), an execution error will occur and it will not be set:

Tw: Pulse width Td: Major delay time Tit: Integration time

# init\_sequence()

This function starts the redirection of write() to store\_sequence\_command() to automatically create a sequence program.

#### start\_sequence(repeat=1)

This function starts the sequence program which is initiated by init\_sequence() and ended by

end\_sequence().

#### end\_sequence()

This function ends the sequence program which is initiated by *init\_sequence()*.

#### sequence\_wait(wait mode, wait value)

Waits for program execution and is used only for sequence programs (WAIT).

#### **Parameters**

- wait\_mode (int) Whether wait time (1) or trigger input count (2) is specified
- wait\_value (float) Wait time or trigger input count as specified by wait\_mode

This command has the following functions:

- Make the execution of the next program wait for the specified time.
- Makes the next program execution wait until the specified number of triggers is input.

Regardless of the wait mode, if the wait data is 0, the wait operation is not performed. When the wait mode is "2", the following commands and signals can be used as trigger inputs:

- XE (XE 0, XE 1, XE 2)
- \*TRG
- GET command (group execute trigger)
- · Trigger input signal on rear panel

# start\_sequence\_program(start, stop, repeat)

Starts from the program number until the stop of the sequence program (RU). Executes sequentially up to the program number, and repeats for the number of times of specified.

# **Parameters**

- **start** (*int*) Number of the program to start from ranging 1 to 100
- **stop** (*int*) Number of the program to stop at ranging from 1 to 100
- **repeat** (*int*) Number of times repeated from 1 to 100

# store\_sequence\_command(line, command)

Stores the program to be executed in the sequence program (ST). If the program already exists, it is replaced with the new sequence.

#### **Parameters**

- line (int) Line number specified of memory location
- **command** (str) Command(s) specified to be stored delimited by a semicolon (;)

# interrupt\_sequence\_command(action)

Interrupts the sequence program executed by the start\_sequence\_program() command (SQSP).

#### **Parameters**

action (SequenceInterruptionType) - Specifies sequence interruption setup

# property sequence\_program\_number

Measure the amount of program sequences stored in the sequence memory (LNUB?).

#### sequence\_program\_listing(line)

This is a query command to know the command list stored in the program number of the sequence program memory (LST?).

#### **Parameters**

**action** (*int*) – Specifying the memory location for reading the commands

#### Returns

Commands stored in sequence memory

#### Return type

str

# trigger\_output\_signal(trigger\_output, alarm\_output, scanner\_output)

Directly output the trigger output signal, alarm output signal, scanner (start/stop) output signal from GPIB (OSIG).

#### **Parameters**

- **trigger\_output** (*int*) Number specifying type of trigger output
- alarm\_output (int) Number specifying type of alaram output
- **scanner\_output** (*int*) Number specifying the type of scanner output

# Trigger output:

- 1. Do not output to trigger output.
- 2. Output a negative pulse to the trigger output.

# Alarm output:

- 1. Finish output GO, LO.HI both set to HI level. (reset)
- 2. Finish output Set GO to LO level.
- 3. Set home output LO to LO level.
- 4. Terminate output HI to LO level.

#### Scanner - (start/stop) output:

- 1. Set the scanner scoot output to HI level. Output a negative pulse to the stop output.
- 2. Make the scanner start output low.
- 3. Output a HI level for the scanner start output and a negative pulse for the stop output.

# set\_output\_format(delimiter\_format, block\_delimiter, terminator)

Sets the format and terminator of the output data output by GPIB (FMT).

# **Parameters**

- **delimiter\_format** (*int*) Type of delimiter format
- block\_delimiter (int) Type of block delimiter
- **terminator** (*int*) Type of termination character

The output of <EOI> (End or Identify) is output at the following timing: 1,2: Simultaneously with LF 4: Simultaneously with the last output data

If the output data format is specified as binary format, the terminator is fixed to <EOI> only and the terminator selection is ignored.

delimiter\_format:

- 1. ASCII format with header
- 2. No header, ASCII format
- 3. Binary format

# block\_delimiter:

- 1. Make it the same as the terminator.
- 2. Use semicolon;
- 3. Use comma,

# terminator:

- 1. CR, LF<EOI>
- 2. LF<EOI>
- 3. LF
- 4. <EOI>

1st character header:			
A)	Normal measurement data  Measurement data during overrange		
B)			
C)	Compliance (limiter) is working.		
D)	Oscillation detection is working.		
E)	[Indicates the generated data]		
F)	Measurement data when an error occurs in the search measurement		
26)	Measurement data is not stored in the buffer memory.		

2nd character header:		
A)	A-channel data during asynchronous operation (A-channel generation data)	
B)	B-channel data during asynchronous operation (channel generation data)	
I)	A-channel data for synchronous, sweeping, de- layed sweep, and double synchronous sweep op- erations.	
J)	B-channel data for synchronous, sweeping, de- layed sweep, and double synchronous sweep op- erations.	

3rd character header:				
A)	Current generation, voltage measurement (ISVM) [Current generation]			
B)	Voltage generation, current measurement (VSIM) [Voltage generation]			
C)	Current generation, current measurement (ISIM)			
D)	Voltage generation, voltage measurement (VSVM Current generation, external voltage measuremen (IS, EXT, VM)			
E)				
F)	Voltage generation, external current measurement (VS, EXT, IM)			
G)	Current generation, external current measuremen (IS, EXT. IM)			
H)	Voltage generation, external voltage measurement (VS, EXT, VM)			
26)	The measurement data is not stored in the buffer memory.			

4th character header:			
A)	No operation (fixed to A)  Null operation result		
B)			
C)	The result of the comparison operation is GO.		
D)	The result of the comparison operation is LO.		
E)	The result of the comparison operation is HI.		
F)	The result of null operation + comparison operation is GO.  The result of null operation + comparison operation is LO.		
G)			
H)	The result of null operation + comparison operation is HI.		
26)	Measurement data is not stored in the buffer memory.		

# property service\_request\_enable\_register

Control the contents of the service request enable register (SRER) in the form of a SRER IntFlag (\*SRE).



# 1 Note

Bits other than the RQS bit are not cleared by serial polling. When power\_on\_clear() is set, status byte enable register, SESER, device operation enable register, channel operation, the enable register is cleared and no SRQ is issued.

# property event\_status\_enable

Control the standard event status enable. (\*ESE)

# property power\_on\_clear

Control the power on clear flag, takes values True or False. (\*PSC)

# property device\_operation\_enable\_register

Control the device operation output enable register (DOER) (DOE?).

# property digital\_out\_enable\_data

Control the contents of digital out enable data set (DIOE).

#### property status\_byte\_register

Measure the contents of the status byte register and MSS bits without using a serial poll (\*STB?).

The Status Byte Register has a hierarchical structure. ERR, DOP, ESB, and COP bits, except RQS and MAV, have lower-level status registers. Each register is paired with an enable register that can be selected to output to the Status Byte register or not. The status byte register also has an enable register, which allows you to select whether or not to issue a service request SRQ.



\*STB? command can read bit 6 as MSS (logical OR of other bits).

# property event\_status\_register

Measure the contents of the standard event status register (SESR) in the form of a SESR IntFlag (\*ESR?).



SESR is cleared after being read.

# property device\_operation\_register

Measure the contents of the device operations register (DOR) in the form of a DOR IntFlag (DOC?).

#### property error\_register

Measure the contents of the error register (ERR?).

#### property self\_test

Get the result of self test after running it.

#### property trigger\_link\_function\_enabled

Set a boolean that controls whether the trigger link function is enabled, takes values of True or False. (TLNK)

Type bool

#### property display\_enabled

Set a boolean that controls whether the display is on or off, takes values of True or False. (DISP)

Type bool

#### property line\_frequency

Set the used power supply frequency (LF) to 50 or 60hz. With this command, the integration time per PLC for the measurement will be one cycle of the power supply frequency you are using.

Type int

# property store\_config

Set the memory area for the config to be stored at (SAV). There are five memory areas from 0 to 4 for storing.

Type int

# property load\_config

Set the memory area for the config to be loaded from (RCL). There are five areas (0~4) where parameters can be loaded by the RCL command.

Type

int

#### set\_lo\_common\_connection\_relay(enable, lo\_relay=None)

Turn the connection relay on/off between the A channel LO (internal analog common) and the LO (internal analog common) of the B channel (LTL).

#### **Parameters**

- **enable** (*boo1*) A boolean property that controls whether or not the connection relay is enabled. Valid values are True and False.
- **lo\_relay** (*bool*, *optional*) A boolean property that controls whether or not the internal analog common relay is enabled. Valid values are True, False and None (don't change lo relay setting).

# read\_measurement()

Reads the triggered value, for example triggered by the external input.

Bases: Channel

Instantiated by main instrument class for every SMUChannel

#### insert\_id(command)

Insert the channel id in a command replacing placeholder.

Subclass this method if you want to do something else, like always prepending the channel id.

# clear\_measurement\_buffer()

Clears the measurement data buffer (MBC).

set\_output\_type(output\_type, measurement\_type)

Sets the output method and type of the GPIB output (OFM).

# **Parameters**

- **output\_type** (int or *OutputType*) A property that controls the type of output
- measurement\_type (int or MeasurementType) A property that controls the measurement type



For the format of the output data, refer to *AdvantestR624X.set\_output\_format()*. For DC and pulse measurements, the output method is fixed to '1' (real-time output). When the output method '3' (buffering output) is specified, the measured data is not stored in memory.

# property analog\_input

Set the analog input terminal (ANALOG INPUT) on the rear panel ON or OFF (FL).

# Type

int

- 1. Turn off the analog input.
- 2. Analog input ON, gain x1.
- 3. Analog input ON, gain x2.5.

#### property trigger\_output\_timing

Set the timing of the trigger output signal output from TRIGGER OUT on the rear panel (TOT). the status in the form of a *TriggerOutputSignalTiming* IntFlag.

#### **Type**

TriggerOutputSignalTiming

# set\_scanner\_control(output, interlock)

Sets the SCANNER CONTROL (START, STOP) output signal and INTERLOCK input signal on the rear panel (SCT).

#### **Parameters**

- **output** (*int*) A property that controls the scanner output
- **interlock** (*int*) A property that controls the scanner interlock type

# output:

- 1. Scanner, Turn off the control signal output.
- 2. Output to the scanner control signal at the start / stop of the sweep.
- 3. Operate / Standby Scanner, Output to the control signal.

# interlock:

- 1. Turn off the interlock signal input.
- 2. Set as a stamper when the interlock signal input is HI.
- 3. When the interlock signal input is HI, it is on standby, and when it is LO, it is operated.

# property trigger\_input

Set the type of trigger input (TJM).

# Type

TriggerInputType

Trigger input types	1	2	3
*TRG	О	0	X
XE 0	O	Ο	X
XE Channel	O	Ο	O
GET	O	Ο	X
Trigger input signal	0	X	X

O can be used, X cannot be used



The sweep operation cannot be started by the trigger input signal. Be sure to start it with the 'XE' command. Once started, it is possible to advance the sweep with a trigger input signal.

# property fast\_mode\_enabled

Set the channel response mode to fast or slow, takes values of True or False (FL).

# Type

bool

#### property sample\_hold\_mode

Set the integration time of the measurement (MST).

#### Type

SampleHold

# **1** Note

- Valid only for pulse measurement and pulse sweep measurement.
- In sample hold mode, the AD transformation is just before the fall of the pulse width.
- The sample hold mode cannot be set during DC measurement and DC sweep measurement. When set to sample-and-hold mode, the integration time is 100 µs. However, in 2-channel synchronous operation, if one channel is in pulse generation and the other is in sample-and-hold mode, the DC measurement side also operates in sample-and-hold mode.
- When performing pulse measurement and pulse sweep measurement, it is necessary to satisfy the restrictions on the pulse width (Tw), pulse period (Tp), and measure delay time (Td) of the WT command. If the constraint is not satisfied, the integration time is unchanged. To lengthen the integration time, first change the pulse width (Tw) and pulse period (Tp). When shortening the pulse width and pulse cycle, shorten the integration time first.

# set\_sample\_mode(mode, auto\_sampling=True)

Sets synchronous, asynchronous, tracking operation and search measurement between channels (JM).

#### **Parameters**

- mode (SampleMode) Sample Mode
- auto\_sampling (bool, optional) Whether or not auto sampling is enabled, defaults to True

set\_timing\_parameters(hold\_time, measurement\_delay, pulsed\_width, pulsed\_period)

Set the hold time, measuring time, pulse width and the pulse period (WT).

# **Parameters**

- hold\_time (float) total amount of time for the complete pulse, until next pulse comes
- measurement\_delay (float) time between measurements
- pulsed\_width (float) Time specifying the pulse width
- pulsed\_period (float) Time specifying the pulse period

#### Note

Pulse measurement has the following restrictions depending on the pulse period (Tp) setting. (For pulse sweep measurements, there are no restrictions.)

- Tp < 2ms : Not measured.
- $2ms \le Tp \le 10ms$ : Measure once every  $5 \sim 20ms$ .
- 10ms <= Tp: Measured at each pulse generation.

#### select\_for\_output()

This is a query command to select a channel and to output the measurement data (FCH?). When the output channel is selected by the FCH command, the measured data of the same channel is returned until the output channel is changed by the next FCH command.

# 1 Note

Reading measurements with the RMM command does not affect channel specification with the FCH command. In the default state, the measurement data of channel A is output.

# trigger()

Measurement trigger command for sweep, start search measurement or sweep step action (XE).

measure\_voltage(enable=True, internal\_measurement=True, voltage\_range=VoltageRange.AUTO)

Sets the voltage measurement ON/OFF, measurement input, and voltage measurement range as parameters (RV).

#### **Parameters**

- enable (bool, optional) boolean property that enables or disables voltage measurement. Valid values are True (Measure the voltage flowing at the OUTPUT terminal) and False (Measure the voltage from the rear panel -ANALOG COMMON).
- internal\_measurement (bool, optional) A boolean property that enables or disables the internal measurement.
- voltage\_range (VoltageRange, optional) Specifying voltage range

voltage\_source(source\_range, source\_value, current\_compliance)

Sets the source range, source value and the current compliance for the DC (constant voltage) measurement (DV).

#### **Parameters**

- **source\_range** (*VoltageRange*) Specifying source range
- source\_value (float) A number specifying the source voltage value
- current\_compliance (float) A number specifying the current compliance



# 1 Note

Regardless of the specified current compliance polarity, both polarities (+ and -) are set. The current compliance range is automatically set to the minimum range that includes the set value.

voltage\_pulsed\_source(source\_range, pulse\_value, base\_value, current\_compliance)

Sets the source range, pulse value, base value and the current compliance of the pulse (voltage) measurement (PV).



# 1 Note

Regardless of the specified current compliance polarity, both polarities (+ and -) are set. The current compliance range is automatically set to the minimum range that includes the set value.

# property change\_source\_voltage

Set new target voltage (SPOT).

# Type

float

# 1 Note

Only the DC action source value and pulse action pulse value are changed using the currently set DC action and pulse action parameters. Measure after the change and set the channel to output the measured data to the specified ch. In other words, it's the same as running the following commands:

- 1. DV/DI/PV/PI
- 2. XE xx
- 3. FCH xx

Sets the fixed level sweep (voltage) generation range, level value, current compliance and the bias value (FXV).

# 1 Note

Regardless of the specified current compliance polarity, both polarities (+ and -) are set. The current compliance range is automatically set to the minimum range that includes the set value.

**voltage\_fixed\_pulsed\_sweep**(voltage\_range, pulse, base, measurement\_count, current\_compliance, bias=0)

Sets the fixed pulse (voltage) sweep generation range, pulse value, base value, number of measurements, current compliance and the bias value (PXV).

# 1 Note

Regardless of the specified current compliance polarity, both polarities (+ and -) are set. The current compliance range is automatically set to the minimum range that includes the set value.

**voltage\_sweep**(sweep\_mode, repeat, voltage\_range, start\_value, stop\_value, steps, current\_compliance, bias=0)

Sets the sweep mode, number of repeats, source range, start value, stop value, number of steps, current compliance, and the bias value for staircase (linear/log) voltage sweep (WV).

# **1** Note

- Sweep mode, number of repeats, and number of steps are subject to the following restrictions.
  - Let N = number of steps, m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the OFM command sets the output data output method to 1 or 2 m x number of refreshes x N <= 2048

- \* m x N  $\leq$  2048 when the OFM command sets the output data output method to 3.
- Regardless of the specified current compliance polarity, both polarities (+ and -) are set.
- The current compliance range is automatically set to the minimum range that includes the set value.

Sets the sweep mode, repeat count, generation range, base value, start value, stop value, number of steps, current compliance and the bias value for a pulse wave (linear/log) voltage sweep (PWV).

# 1 Note

- The sweep mode, number of refreshes, and number of steps are subject to the following restrictions:
  - Let N = number of steps, m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the OFM command sets the output data output method to 1 or 2 m x number of refreshes x N <= 2048
    - \* m x N  $\leq$  2048 when the OFM command sets the output data output method to 3.
- For the current compliance polarity, regardless of the specified current compliance polarity, the compliance of both polarities (+ and -) is set.
- The current compliance range is automatically set to the minimum range that includes the set value.

**voltage\_random\_sweep**(*sweep\_mode*, *repeat*, *start\_address*, *stop\_address*, *current\_compliance*, *bias=0*)

Sets the sweep mode, repeat count, start address, stop address, current compliance and the bias value of constant voltage random sweep (MDWV).

#### 1 Note

- Sweep mode, number of repeats, start address and stop address are subject to the following restrictions:
  - Start address < Stop address
  - Let N = number of steps, m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the OFM command sets the output data output method to 1 or 2 m x number of refreshes x N  $\leq$  2048
    - \* m x N  $\leq$  2048 when the OFM command sets the output data output method to 3.
- Regardless of the specified current compliance polarity, both polarities (+ and -) are set.
- The current compliance range is automatically set to the minimum range that includes the set value.

Sets the sweep mode, repeat count, base value, start address, stop address, current compliance and the bias

value of the constant voltage random pulse sweep (MPWV).

# **1** Note

- Sweep mode, number of repeats, start address and stop address are subject to the following restrictions:
  - Start address < Stop address
  - Let N = number of steps, m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the OFM command sets the output data output method to 1 or 2 m x number of refreshes x N  $\leq$  2048
    - \*  $m \times N \le 2048$  when the OFM command sets the output data output method to 3.
- Regardless of the specified current compliance polarity, both polarities (+ and -) are set.
- The current compliance range is automatically set to the minimum range that includes the set value.

# **voltage\_set\_random\_memory**(address, voltage\_range, output, current\_compliance)

The command stores the specified value to the randomly generated data memory (RMS).

Stored generated values are swept within the specified memory address range by the MDWV, MDWI, MPWV, MPWI commands.

current\_source(source\_range, source\_value, voltage\_compliance)

Sets the source range, source value, voltage compliance of the DC (constant current) measurement (DI).

# Parameters

- source\_range (CurrentRange) Specifying source range
- source\_value (float) A number specifying the source current value
- voltage\_compliance (float) A number specifying the voltage compliance

# 1 Note

Regardless of the specified voltage compliance polarity, both polarities (+ and -) are set. The voltage compliance range is automatically set to the minimum range that includes the set value.

# current\_pulsed\_source(source\_range, pulse\_value, base\_value, voltage\_compliance)

Sets the source range, pulse value, base value and the voltage compliance of the pulse (current) measurement (PI).

# 1 Note

Regardless of the specified voltage compliance polarity, both polarities (+ and -) are set. The voltage compliance range is automatically set to the minimum range that includes the set value.

#### property change\_source\_current

Set new target current (SPOT).

# **Type**

float

#### 1 Note

Only the DC action source value and pulse action pulse value are changed using the currently set DC action and pulse action parameters. Measure after the change and set the channel to output the measured data to the specified ch. In other words, it's the same as running the following commands:

- 1. DV/DI/PV/PI
- 2. XE xx
- 3. FCH xx

current\_fixed\_level\_sweep(current\_range, current\_level, measurement\_count, voltage\_compliance, bias=0)

Sets the fixed level sweep (current) generation range, level value, voltage compliance and the bias value (FXI).

# 1 Note

Regardless of the specified voltage compliance polarity, both polarities (+ and -) are set. The voltage compliance range is automatically set to the minimum range that includes the set value.

current\_fixed\_pulsed\_sweep(current\_range, pulse, base, measurement\_count, voltage\_compliance, bias=0)

Sets the fixed pulse (current) sweep generation range, pulse value, base value, number of measurements, voltage compliance and the bias value (PXI).



# 1 Note

Regardless of the specified voltage compliance polarity, both polarities of + and - are set. The voltage compliance range is automatically set to the minimum range that includes the set value.

current\_sweep(sweep\_mode, repeat, current\_range, start\_value, stop\_value, steps, voltage\_compliance, bias=0)

Sets the sweep mode, number of repeats, source range, start value, stop value, number of steps, voltage compliance and bias value for the staircase (linear/log) current sweep (WI).

#### 1 Note

- The sweep mode, number of refreshes, and number of steps are subject to the following restric-
  - Let N = number of steps, m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the OFM command sets the output data output method to 1 or 2, m x number of repeats  $x N \le 2048$ .
    - \* m x N  $\leq$  2048 when the OFM command sets the output data output method to 3.

- Regardless of the specified voltage compliance polarity, both polarities (+ and -) are set.
- The voltage compliance range is automatically set to the minimum range that includes the set value.

**current\_pulsed\_sweep**(sweep\_mode, repeat, current\_range, base, start\_value, stop\_value, steps, voltage\_compliance, bias=0)

Sets the sweep mode, repeat count, generation range, base value, start value, stop value, number of steps, voltage compliance and the bias value for a pulse wave (linear/log) current sweep (PWI).

# **1** Note

- The sweep mode, number of refreshes, and number of steps are subject to the following restrictions:
  - Let N = number of steps, m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the OFM command sets the output data output method to 1 or 2, m x number of repeats  $x N \le 2048$ .
    - \* m x N  $\leq$  2048 when the OFM command sets the output data output method to 3.
- Regardless of the specified voltage compliance polarity, both polarities (+ and -) are set.
- The voltage compliance range is automatically set to the minimum range that includes the set value.

measure\_current(enable=True, internal measurement=True, current range=CurrentRange.AUTO)

Set the current measurement ON/OFF, measurement input, and current measurement range as parameters (RI).

#### **Parameters**

- **enable** (*bool*, *optional*) boolean property that enables or disables current measurement. Valid values are True (Measure the current flowing at the OUTPUT terminal) and False (Measure the current from the rear panel -ANALOG COMMON).
- internal\_measurement (bool, optional) A boolean property that enables or disables the internal measurement.
- **current\_range** (*CurrentRange*, optional) Specifying voltage range

**current\_random\_sweep**(*sweep\_mode*, *repeat*, *start\_address*, *stop\_address*, *current\_compliance*, *bias=0*)

Sets the sweep mode, repeat count, start address, stop address, voltage compliance and the bias value of constant current random sweep (MDWI).

# 1 Note

- Sweep mode, number of repeats, start address and stop address are subject to the following restrictions:
  - Start address < Stop address</li>
  - Let N = (stop number 1 start number + 1), m = 1 (one-way sweep), m = 2 (round-trip sweep).

- \* When the output data output method is set to 1 or 2 with the OFM command m x number of repeats x N  $\leq$  2048
- \* When the output data output method is set to 3 with the OFM command m x N  $\leq$  2048
- For the voltage compliance polarity, regardless of the specified voltage compliance polarity, both polarities of + and are set.
- The voltage compliance range is automatically set to the minimum range that includes the set value.

Sets the sweep mode, repeat count, base value, start address, stop address, voltage compliance and the bias value of constant current random pulse sweep (MPWI).

# 1 Note

- Sweep mode, number of repeats, start address and stop address are subject to the following restrictions:
  - Start address < Stop address</li>
  - Let N = (stop number 1 start number + 1), m = 1 (one-way sweep), m = 2 (round-trip sweep).
    - \* When the output data output method is set to 1 or 2 with the OFM command m x number of repeats x N  $\leq$  2048
    - \* When the output data output method is set to 3 with the OFM command m x N  $\leq$  2048
- For the voltage compliance polarity, regardless of the specified voltage compliance polarity, both polarities of + and are set.
- The voltage compliance range is automatically set to the minimum range that includes the set value

current\_set\_random\_memory(address, current\_range, output, voltage\_compliance)

Store the current parameters to randomly generated data memory (RMS).

Stored generated values are swept within the specified memory address range by the MDWV, MDWI, MPWV, MPWI commands.

#### read\_random\_memory(address)

Return memory specified by address location (RMS?).

#### **Parameters**

**address** (*int*) – Adress to specify memory location.

#### Returns

Set values returned by the device from the specified address location.

#### Return type

str

#### enable\_source()

Put the specified channel into an operating state (CN).

#### standby()

Put the specified channel into standby state (CL).

#### stop()

Stops the sweep when the sweep is started by the XE command or the trigger input signal (SP).

#### output\_all\_measurements()

Output all measurements in the measurement data buffer of the specified channel (RMM?).

# 1 Note

For the output format, refer to *AdvantestR624X*. *set\_output\_format()*. When a memory address where no measurement data is stored is read, 999.999E+99 will be returned.

#### read\_measurement\_from\_addr(addr)

Output only one measurement at the specified memory address from the measurement data buffer of the specified channel.

#### **Parameters**

**addr** (*int*) – Specifies the address to read from.

#### **Returns**

float Measurement data

# 1 Note

For the output format, refer to *AdvantestR624X.set\_output\_format()*. When a memory address where no measurement data is stored is read, 999.999E+99 will be returned.

# property measurement\_count

Measure the number of measurements contained in the measurement data buffer (NUB?).

#### property null\_operation\_enabled

Set a boolean that controls whether the null operation is enabled, takes values of True or False (NUG).

# **Type**

bool

# **1** Note

- Null data is not rewritten even if the null operation is disabled.
- Null data is rewritten only when null operation is changed from OFF to ON or initialized in case of DC operation or pulse operation.

# set\_wire\_mode(four\_wire, lo\_guard=True)

Used to switch remote sense and to set the LO-GUARD relay ON/OFF. It operates regardless of operating state or standby state (OSL).

# **Parameters**

- **four\_wire** (*bool*) A boolean property that enables or disables four wire measurements. Valid values are True (enables 4-wire sensing) and False (enables two-terminal sensing).
- **lo\_guard** (*bool*) A boolean property that enables or disables the LO-GUARD relay.

# property auto\_zero\_enabled

Set the auto zero option to ON or OFF. Valid values are True (enabled) and False (disabled) (CM).

#### Type

bool

This command sets auto zero (automatically calibrate the zero point of the measured value operation.

- 1. Periodically perform auto zero.
- 2. Auto zero once, no periodic auto zeros thereafter.

When the auto zero mode is set to True, the following operations are performed.

- For DC operation and pulse operation:
  - At the end of one sweep, if he has exceeded the last autozero by more than 10 seconds, he will do
    one autozero.
  - If sweep start is specified during auto zero, the sweep will start after auto zero ends.
- Sweep operation
  - Auto zero is performed once every 10 seconds.
  - If measurement or pulse output is specified during auto zero, it will be executed after auto zero ends.

#### set\_comparison\_limits(comparison, voltage\_value, upper\_limit, lower\_limit)

Sets the channel ON/OFF based on the measurement comparison and the data of the upper and lower limits to be compared (CMD).

#### **Parameters**

- **comparison** (*boo1*) A boolean property that controls whether or not the comparison function is enabled. Valid values are True or False.
- **voltage\_value** (*bool*) A boolean property that controls whether or not voltage or current values are passed. Valid values are True or False.
- upper\_limit (float) Number specifying the upper comparison limit
- lower\_limit (float) Number specifying the lower comparison limit

#### property relay\_mode

Set the HI/LO relays for standby mode. This command does not operate the Operate Relay (OPM).

#### Type

int

- 1. When executing an operation only the HI side turns ON, in standby both HI and LO are turned OFF.
- 2. When executing an operation only the LO side turns ON, in standby both HI and LO are turned OFF.
- 3. When executing an operation both HI and LO turn ON, in standby both HI and LO are turned OFF.
- 4. When executing an operation only the HI side turns ON, in standby only the HI side is turned OFF.

### property operation\_register

Measure the contents of the Channel Operations Register (COR) in the form of a COR IntFlag (COC?).

# property output\_enable\_register

Control the settings of the channel operation output enable register (COER) in the form of a *COR* IntFlag ?(COE?).

## calibration\_init()

Initialize the calibration data (CINI).

### calibration\_store\_factor()

Store the calibration factor in the non-volatile memory (EEPROM) (CSRT).

# property calibration\_measured\_value

Set the measured value measured by an external standard for the generated value of this instrument and start calibration (STD).

# Type

float

# property calibration\_generation\_factor

Set the increment or decrement for the generation calibration factor of the current generation range (CCS). It is used when the generated value deviates from the true value.

### Type

float

# property calibration\_factor

Set the increment of the measurement calibration factor of the current measurement range (CCM).

## Type

float

\*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntEnum

qualname=None, type=None, start=1, boundary=None)

Bases: IntEnum

\*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntEnum

class pymeasure.instruments.advantest.advantestR624X.CurrentRange(value, names=<not given>,

\*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntEnum

7.8. Advantest

class pymeasure.instruments.advantest.advantestR624X.SweepMode(value, names=<not given>,

```
*values, module=None,
                                                                          qualname=None, type=None,
                                                                          start=1, boundary=None)
     Bases: IntEnum
class pymeasure.instruments.advantest.advantestR624X.OutputType(value, names=<not given>,
                                                                           *values, module=None,
                                                                           qualname=None, type=None,
                                                                           start=1, boundary=None)
     Bases: IntEnum
class pymeasure.instruments.advantest.advantestR624X.TriggerInputType(value, names=<not</pre>
                                                                                  given>, *values,
                                                                                  module=None,
                                                                                  qualname=None,
                                                                                  type=None, start=1,
                                                                                  boundary=None)
     Bases: IntEnum
class pymeasure.instruments.advantest.advantestR624X.MeasurementType(value, names=<not</pre>
                                                                                 given>, *values,
                                                                                 module=None,
                                                                                 qualname=None,
                                                                                 type=None, start=1,
                                                                                 boundary=None)
     Bases: IntEnum
class pymeasure.instruments.advantest.advantestR624X.SequenceInterruptionType(value,
                                                                                           names=<not
                                                                                           given>,
                                                                                           *values,
                                                                                           module=None,
                                                                                           qual-
                                                                                           name=None,
                                                                                           type=None,
                                                                                           start=1,
                                                                                           bound-
                                                                                           ary=None)
     Bases: IntEnum
       1. Release pause state is a valid command only in the sequence program pause state. otherwise it is ignored.
       2. Pause state enters the pause state when the currently executing program ends.
       3. Abort sequence program stops the sequence program when the currently executing program ends. If the
          currently running program is a sweep operation, interrupt the sweep operation and stop the sequence pro-
          gram. The output value will be the bias value.
```

class pymeasure.instruments.advantest.advantestR624X.DOR(value, names=<not given>, \*values,

bit assigment for the Device Operation Register (DOR):

module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntFlag

Bit (dec)	Description
13	Indicates that the fast tokens program is running.
12	Error in search measurement
11	End of sequence program/high-speed sequence program execution
10	Sequence program Pause state
9	Fan stop detection
8	Self-test error occurred (logic part)
7	Trigger wait state in trigger link master operation
6	Calibration mode status
5	Trigger link ON state
4	Trigger link bus error
3	Sequence program/high-speed sequence 1 program/add/de) waiting
2	Wait for sequence program wait time
1	Sequence program running
0	Synchronous operation state

 $\textbf{class} \ \ \textbf{pymeasure.instruments.advantest.advantestR624X.COR} (\textit{value}, \textit{names} = < \textit{not given} >, *\textit{values}, \\ \textit{module} = \textit{None}, \textit{qualname} = \textit{None}, \\ \textit{type} = \textit{None}, \textit{start} = 1, \textit{boundary} = \textit{None})$ 

Bases: IntFlag

bit assignment for the Channel Operations Register (COR):

Bit (dec)	Description
14	The result of the comparison operation is HI
13	The result of the comparison operation is GO
12	The result of the comparison operation is LO
11	Overheat detection
10	Overload detection
9	Oscillation detection
8	Compliance detection
7	Synchronous operation master channel
6	Measurement data output specification
5	There is measurement data
4	Self-test error occurrence (analog part)
3	Measurement data buffer full
2	Waiting for trigger
1	End of sweep
0	Operated state

Bases: IntFlag

bit assigment for the Service Request Enable Register (SRER):

7.8. Advantest

Bit (dec)	Description
0	none
1	ERR Set when any of QYE, DDE, EXE, or CME in the Standard Event Status Register (SESR) is set.
2	DOP Set when a bit in the device operation register for which the enable register is set to enabled is set. Cleared by reading the device operation register.
3	none
4	MAV Set when output data is set in the output queue. Cleared when output data is read.
5	ESB Set when a bit in the Standard Event Status Register (SESR) is set and the enable register is set to Enabled. Cleared by reading SESR.
6	RQS (MSS) Set when bit O to bit 5 and bit 7 of the Status Byte register are set. (this bit is read-only)
7	COP Set when a bit in the Channel Operations Register is set with the Enable Register set to Enable. Cleared by reading the Channel Operations Register.

Bases: IntFlag

bit assigment for the Standard Event Status Register (SESR):

Bit (dec)	Description
0	OPC (Operation Complete) not used
1	RQC unused
2	QYE (Query Error) Set when the output queue overflows when reading without output data.
3	DDE (Device Dependent Error) Set when an error occurs in the self-test.
4	EXE (Execution Error) Set when the input data is outside the range set internally, or when the command cannot be executed.
5	CME (Command Error) Set when an undefined header or data format is wrong, or when there is a syntax error in the command.
6	URQ unused
7	PON Set when power is switched from OFF to ON.

class pymeasure.instruments.advantest.advantestR624X.TriggerOutputSignalTiming(value,

names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntFlag

bit assignment for the timing of the trigger output signal output from TRIGGER OUT on the rear panel:

Bit (dec)	Description
5	At the end of the sweep
4	At the end of the pulse width
3	At the end of the pulse cycle
2	At the end of measurement
1	At the start of measurement
0	At the start of occurrence

# **Contents**

- Advantest R6245/R6246 DC Voltage/Current Sources/Monitors
  - Main Classes
  - General Information
  - Examples
    - \* Initialization of the Instrument
    - \* Simple dual channel measurement example
    - \* Program example for DC measurement
    - \* Program example for DC measurement (with external trigger)
    - \* Program example for pulse measurement
    - \* Fixed Level Sweep Program Example

#### **General Information**

The R6245/6246 Series are DC voltage/current sources and monitors having source measurement units (SMUs) with 2 isolated channels. The series covers wide source and measurement ranges. It is ideal for measurement of DC characteristics of items ranging from separate semiconductors such as bipolar transistors, MOSFETs and GaAsFETs, to ICs and power devices. Further, due to the increased measuring speed and synchronized 2-channel measurement function, device I/O characteristics can be measured with precise timing at high speed which was previously difficult to accomplish. Due to features such as the trigger link function and the sequence programming function which automatically performs a series of evaluation tests automatically, the R6245/6246 enable much more efficient evaluation tests.

There is a total of 99 commands, the majority of commands have been implemented. Device documentation is in Japanese, and the device options are enormous. The implementation is based on 6245S-GPIB-B-FHJ-8335160E01.pdf, which can be downloaded from the ADCMT website.

7.8. Advantest 143

# **Examples**

#### Initialization of the Instrument

```
from pymeasure.instruments.advantest import AdvantestR6246
from pymeasure.instruments.advantest.advantestR624X import *
smu = AdvantestR6246("GPIB::1")
```

# Simple dual channel measurement example

## **Measurement characteristics:**

Channel A: Vce = 20V Channel B: Ib = 10uA - 60uA

```
smu = AdvantestR6246("GPIB::1")
                                                            # Set default parameters
smu.reset()
smu.ch_A.set_sample_mode(SampleMode.PULSED_SYNC)
                                                            # Pulsed synchronized
smu.ch_A.voltage_source(source_range = VoltageRange.AUTO,
                        source_value = 20,
                        current_compliance = 0.06)
smu.ch_A.measure_current()
smu.ch_B.current_source(source_range = CurrentRange.AUTO,
                        source_value = 1E-5,
                                                            # Source current at 10 uA
                        voltage_compliance = 5)
                                                           # Voltage compliance at 5 V
smu.ch_B.measure_voltage()
smu.enable_source()
                                                            # Enables source A & B
for i in range(10, 60):
    k = i * 0.000001
                                                            # Set current from 10 uA to 60.
    smu.ch_B.current_change_source = k
\hookrightarrow uA
                                                            # Trigger measurement
    smu.trigger()
    smu.ch_A.select_for_output()
    Ic = smu.read_measurement()
                                                            # Read channel A measurement
    smu.ch_B.select_for_output()
    Vbe = smu.read_measurement()
                                                            # Read channel B measurement
    print(f'Ic={Ic}, Vbe={Vbe}')
                                                            # Print measurements
smu.standby()
                                                            # Put channel A & B in standby
```

### Program example for DC measurement

### **Measurement characteristics:**

Function: VSIM - Source voltage and measure current Trigger voltage: 10V Current compliance: 0.5A Measurement delay time: 1ms Integration time: 1 PLC Response: Fast

After operating, the measurement is repeated 10 times with a trigger command and he prints out the results.

```
smu = AdvantestR6246("GPIB::1")
smu.reset()
                                                                   # Set default parameters
smu.ch_A.set_sample_mode(SampleMode.ASYNC, False)
                                                                   # Asynchronous_
→operation and single shot sampling by trigger and command
smu.ch_A.voltage_source(source_range = VoltageRange.FIXED_BEST,
                        source_value = 10,
                                                                  # compliance of 0.5A
                        current_compliance = 0.5)
                                                                  # Measure current
smu.ch_A.measure_current()
smu.ch_A.set_timing_parameters(hold_time = 0,
                                                                  # 0 sec hold time
                               measurement_delay = 1E-3,
                                                                  # 1ms delay between_
→ measurements
                               pulsed\_width = 5E-3,
                                                                  # 5ms pulse width
                               pulsed_period = 10E-3)
                                                                  # 10ms pulse period
smu.ch_A.sample_hold_mode = SampleHold.MODE_1PLC
                                                                  # Sample at 1 power_
→line cycle
smu.ch_A.fast_mode_enabled = True
                                                                   # Set channel response.
→to fast
smu.ch A.enable source()
                                                                   # Set channel in...
→operating state
smu.ch_A.select_for_output()
                                                                   # Select channel for...
\hookrightarrow measurement output
for i in range(1, 10):
    smu.ch_A.trigger()
                                                                   # Trigger a measurement
   measurement = smu.read_measurement()
   print(f"NO {i} {measurement}")
smu.ch_A.standby()
                                                                   # Put channel A in_
→standby mode
```

# Program example for DC measurement (with external trigger)

### **Measurement characteristics:**

Function: VSIM - Source voltage and measure current Source voltage: 10 V Base voltage 1 V Current compliance: 0.5 A Pulse width: 5 ms Pulse period: 10 ms Measurement delay time: 1 ms Integration time: 1 ms Response: Fast

After operating, an external trigger input signal is pulsed to measure the channel operation register. Reads the fixed end bit, captures the measurement data, and prints out the measurement result.

7.8. Advantest

```
smu.ch_A.fast_mode_enabled = True
                                                                 # Set channel response.
→to fast
smu.ch_A.sample_hold_mode = SampleHold.MODE_1mS
                                                                 # Sample at 1mS
smu.ch_A.set_timing_parameters(hold_time = 0,
                                                                 # 0 sec hold time
                               measurement_delay = 1E-3,
                                                                 # 1ms delay between
\hookrightarrow measurements
                                                                # 5ms pulse width
                               pulsed\_width = 5E-3,
                               pulsed_period = 10E-3)
                                                               # 10ms pulse period
smu.ch_A.trigger_input = TriggerInputType.ALL
                                                                # Mode 1 enables the
→trigger input signal
smu.ch_A.output_enable_register = COR.HAS_MEASUREMENT_DATA
                                                                # Measurement data_
→available
smu.service_request_enable_register = SRER.COP
                                                                 # COP Set when a bit in_
→the Channel Operations Register is set with the Enable Register set to Enable.
smu.ch_A.enable_source()
                                                                 # Set channel in_
→operating state
                                                                 # Select channel for
smu.ch_A.select_for_output()
→measurement output
for i in range(1, 10):
   while not smu.ch_A.operation_register & COR.HAS_MEASUREMENT_DATA:
   measurement = smu.read_measurement()
   print(f"NO {i} {measurement}")
   while not smu.ch_A.operation_register & COR.WAITING_FOR_TRIGGER:
       pass
smu.ch_A.standby()
                                                                 # Put channel A in_
⇒standby mode
```

## Program example for pulse measurement

### **Measurement characteristics:**

Function: ISVM - Source current and measure voltage Pulse generation current: 100mA Base current: 1mA Voltage compliance: 5V Pulse width: 0 Pulse period: 0 Measurement delay time: 0 Integration time: 1ms Response: Fast

After the operation, repeat the measurement 10 times with the trigger command and print out the measurement results.

(continues on next page)

```
smu.ch_A.fast_mode_enabled = True
                                                                     # Set channel
⇔response to fast
smu.ch_A.sample_hold_mode = SampleHold.MODE_1mS
                                                                     # Sample at 1mS
                                                                     # 0 sec hold time
smu.ch_A.set_timing_parameters(hold_time = 0,
                               measurement_delay = 0,
                                                                    # 0 sec delay.
→between measurements
                                                                    # 0 sec pulse width
                               pulsed\_width = 0,
                               pulsed_period = 0)
                                                                    # 0 sec pulse period
                                                                    # Set channel in
smu.ch_A.enable_source()
→operating state
                                                                     # Select channel for
smu.ch_A.select_for_output()
→measurement output
for i in range(1, 10):
    smu.ch_A.trigger()
                                                                     # Trigger measurement
   measurement = smu.read_measurement()
   print(f"NO {i} {measurement}")
   while not smu.ch_A.operation_register & COR.WAITING_FOR_TRIGGER:
       pass
smu.ch_A.standby()
                                                                     # Put channel A in.
→standby mode
```

## **Fixed Level Sweep Program Example**

## **Measurement characteristics:**

function: VSVM - Voltage source and voltage measurement Level value: 15V Bias value: 0V Number of measurements: 20 times Compliance: 6mA Measuring range: Best fixed range (=60V range) Integration time: 100us Measurement delay time: 0 Hold time: 1ms Sampling mode: automatic sweep Measurement data output method: Buffering output (output of specified data)

After operating, make 20 measurements in fixed sweep. Detect the end of sweep by looking at the Channel Operation Register (COR). After the sweep is finished, read the measured data from 1 to 2 using the RMM command.

```
smu = AdvantestR6246("GPIB::1")
# First we setup our main parameters
smu.reset()
                                                                   # Set default parameters
smu.ch_A.set_output_type(output_type = OutputType.BUFFERING_OUTPUT_SPECIFIED,
                         measurement_type = MeasurementType.MEASURE_DATA)
smu.set_output_format(delimiter_format = 2,
                                                                   # No header, ASCII_
→format
                      block_delimiter = 1,
                                                                   # Make it the same as_
→the terminator
                      terminator = 1)
                                                                   # CR, LF<E0I>
                                                                   # Turn off the analog_
smu.ch_A.analog_input = 1
                                                                             (continues on next page)
```

7.8. Advantest

```
⇔input.
smu.set_lo_common_connection_relay(enable = True)
                                                                # Turns the connection_
→relay on
                                                                # disable four wire.
smu.ch_A.set_wire_mode(four_wire = False,
→measurements
                       lo_guard = True)
                                                                 # enable the LO-GUARD
⊶relay.
smu.ch_A.auto_zero_enabled = False
smu.ch_A.trigger_input = TriggerInputType.ALL
                                                                # Mode 1 enables the
→trigger input signal
# Now we set measurement specific variables
smu.ch_A.clear_measurement_buffer()
smu.ch_A.set_sample_mode(SampleMode.ASYNC, auto_sampling = True)
smu.ch_A.voltage_fixed_level_sweep(voltage_range = VoltageRange.FIXED_60V,
                                   voltage_level = 15,
                                   measurement\_count = 20,
                                                                 # 20 measurements
                                   current_compliance = 6E-3, # compliance at 6mA
                                   bias = 0)
smu.ch_A.measure_voltage(voltage_range = VoltageRange.FIXED_BEST)
smu.ch\_A.sample\_hold\_mode = SampleHold.MODE\_100uS
smu.ch_A.set_timing_parameters(hold_time = 1E-3,
                                                                # 1ms sec hold time
                              measurement_delay = 0,
                                                                # 0 sec delay between_
→measurements
                               pulsed_width = 0,
                                                                # 0 sec pulse width
                               pulsed_period = 0)
                                                                # 0 sec pulse period
smu.ch_A.enable_source()
                                                                 # Set channel in_
→operating state
smu.ch_A.trigger()
                                                                 # Start the sweep
                                                               # Wait until the sweep_
while not smu.ch_A.operation_register & COR.END_OF_SWEEP:
⊶is done
   pass
# Read measurements
for i in range(1, 20):
   measurement = smu.ch_A.read_measurement_from_addr(i)
   print(i, measurement)
                                                                 # Put channel A in_
smu.ch_A.standby()
→standby mode
```

# 7.9 Agilent

This section contains specific documentation on the Agilent instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

If the instrument you are looking for is not here, also check HP for older instruments or Keysight for newer ones.

# 7.9.1 Agilent 8257D Signal Generator

Bases: SCPIUnknownMixin, Instrument

Represents the Agilent 8257D Signal Generator and provides a high-level interface for interacting with the instrument.

# property amplitude\_depth

A floating point property that controls the amplitude modulation in percent, which can take values from 0 to 100 %.

## property amplitude\_source

A string property that controls the source of the amplitude modulation signal, which can take the values: 'internal', 'internal 2', 'external', and 'external 2'.

# property center\_frequency

A floating point property that represents the center frequency in Hz. This property can be set.

```
config_amplitude_modulation(frequency=1000.0, depth=100.0, shape='sine')
```

Configures the amplitude modulation of the output signal.

# **Parameters**

- **frequency** A modulation frequency for the internal oscillator
- **depth** A linear depth percentage
- shape A string that describes the shape for the internal oscillator

```
config_low_freq_out(source='internal', amplitude=3)
```

Configures the low-frequency output signal.

# **Parameters**

- **source** The source for the low-frequency output signal.
- amplitude Amplitude of the low-frequency output

```
config_pulse_modulation(frequency=1000.0, input='square')
```

Configures the pulse modulation of the output signal.

### **Parameters**

• **frequency** – A pulse rate frequency in Hertz

• **input** – A string that describes the internal pulse input

# config\_step\_sweep()

Configures a step sweep through frequency

### disable()

Disables the output of the signal.

### disable\_amplitude\_modulation()

Disables amplitude modulation of the output signal.

### disable\_low\_freq\_out()

Disables low frequency output

### disable\_modulation()

Disables the signal modulation.

### disable\_pulse\_modulation()

Disables pulse modulation of the output signal.

# property dwell\_time

A floating point property that represents the settling time in seconds at the current frequency or power setting. This property can be set.

### enable()

Enables the output of the signal.

### enable\_amplitude\_modulation()

Enables amplitude modulation of the output signal.

## enable\_low\_freq\_out()

Enables low frequency output

## enable\_pulse\_modulation()

Enables pulse modulation of the output signal.

### property frequency

A floating point property that represents the output frequency in Hz. This property can be set.

# property has\_amplitude\_modulation

Reads a boolean value that is True if the amplitude modulation is enabled.

### property has\_modulation

Reads a boolean value that is True if the modulation is enabled.

### property has\_pulse\_modulation

Reads a boolean value that is True if the pulse modulation is enabled.

# property internal\_frequency

A floating point property that controls the frequency of the internal oscillator in Hertz, which can take values from 0.5 Hz to 1 MHz.

#### property internal\_shape

A string property that controls the shape of the internal oscillations, which can take the values: 'sine', 'triangle', 'square', 'ramp', 'noise', 'dual-sine', and 'swept-sine'.

# property is\_enabled

Reads a boolean value that is True if the output is on.

# property low\_freq\_out\_amplitude

A floating point property that controls the peak voltage (amplitude) of the low frequency output in volts, which can take values from 0-3.5V

### property low\_freq\_out\_source

A string property which controls the source of the low frequency output, which can take the values 'internal [2]' for the internal source, or 'function [2]' for an internal function generator which can be configured.

### property power

A floating point property that represents the output power in dBm. This property can be set.

### property pulse\_frequency

A floating point property that controls the pulse rate frequency in Hertz, which can take values from 0.1 Hz to 10 MHz.

# property pulse\_input

A string property that controls the internally generated modulation input for the pulse modulation, which can take the values: 'square', 'free-run', 'triggered', 'doublet', and 'gated'.

## property pulse\_source

A string property that controls the source of the pulse modulation signal, which can take the values: 'internal', 'external', and 'scalar'.

### shutdown()

Shuts down the instrument by disabling any modulation and the output signal.

## property start\_frequency

A floating point property that represents the start frequency in Hz. This property can be set.

## property start\_power

A floating point property that represents the start power in dBm. This property can be set.

# start\_step\_sweep()

Starts a step sweep.

### property step\_points

An integer number of points in a step sweep. This property can be set.

# property stop\_frequency

A floating point property that represents the stop frequency in Hz. This property can be set.

### property stop\_power

A floating point property that represents the stop power in dBm. This property can be set.

### stop\_step\_sweep()

Stops a step sweep.

# 7.9.2 Agilent 8722ES Vector Network Analyzer

Bases: SCPIUnknownMixin, Instrument

Represents the Agilent8722ES Vector Network Analyzer and provides a high-level interface for taking scans of the scattering parameters.

#### property averages

An integer representing the number of averages to take. Note that averaging must be enabled for this to take effect. This property can be set.

# property averaging\_enabled

A bool that indicates whether or not averaging is enabled. This property can be set.

# property data

Returns the real and imaginary data from the last scan

### property data\_complex

Returns the complex power from the last scan

# property data\_log\_magnitude

Returns the absolute magnitude values in dB from the last scan

### property data\_magnitude

Returns the absolute magnitude values from the last scan

### property data\_phase

Returns the phase in degrees from the last scan

# disable\_averaging()

Disables averaging

## enable\_averaging()

Enables averaging

# property frequencies

Returns a list of frequencies from the last scan

# is\_averaging()

Returns True if averaging is enabled

## log\_magnitude(real, imaginary)

Returns the magnitude in dB from a real and imaginary number or numpy arrays

### magnitude(real, imaginary)

Returns the magnitude from a real and imaginary number or numpy arrays

# phase(real, imaginary)

Returns the phase in degrees from a real and imaginary number or numpy arrays

### **scan**(averages=None, blocking=None, timeout=None, delay=None)

Initiates a scan with the number of averages specified and blocks until the operation is complete.

# scan\_continuous()

Initiates a continuous scan

### property scan\_points

Gets the number of scan points

## scan\_single()

Initiates a single scan

# set\_IF\_bandwidth(bandwidth)

Sets the resolution bandwidth (IF bandwidth)

# set\_averaging(averages)

Sets the number of averages and enables/disables averaging. Should be between 1 and 999

# set\_fixed\_frequency(frequency)

Sets the scan to be of only one frequency in Hz

### property start\_frequency

A floating point property that represents the start frequency in Hz. This property can be set.

### property stop\_frequency

A floating point property that represents the stop frequency in Hz. This property can be set.

# property sweep\_time

A floating point property that represents the sweep time in seconds. This property can be set.

# 7.9.3 Agilent E4408B Spectrum Analyzer

Bases: SCPIUnknownMixin, Instrument

Represents the AgilentE4408B Spectrum Analyzer and provides a high-level interface for taking scans of high-frequency spectrums

### property center\_frequency

A floating point property that represents the center frequency in Hz. This property can be set.

#### property frequencies

Returns a numpy array of frequencies in Hz that correspond to the current settings of the instrument.

#### property frequency\_points

An integer property that represents the number of frequency points in the sweep. This property can take values from 101 to 8192.

### property frequency\_step

A floating point property that represents the frequency step in Hz. This property can be set.

### property start\_frequency

A floating point property that represents the start frequency in Hz. This property can be set.

# property stop\_frequency

A floating point property that represents the stop frequency in Hz. This property can be set.

#### property sweep\_time

A floating point property that represents the sweep time in seconds. This property can be set.

### trace(number=1)

Returns a numpy array of the data for a particular trace based on the trace number (1, 2, or 3).

# trace\_df(number=1)

Returns a pandas DataFrame containing the frequency and peak data for a particular trace, based on the trace number (1, 2, or 3).

# 7.9.4 Agilent E4980 LCR Meter

Bases: SCPIUnknownMixin, Instrument

Represents LCR meter E4980A/AL

### property ac\_current

AC current level, in Amps

### property ac\_voltage

AC voltage level, in Volts

aperture(time=None, averages=1)

Set and get aperture.

#### **Parameters**

- **time** integration time as string: SHORT, MED, LONG (case insensitive); if None, get values
- averages number of averages, numeric

freq\_sweep(freq\_list, return\_freq=False)

Run frequency list sweep using sequential trigger.

#### **Parameters**

- **freq\_list** list of frequencies
- return\_freq if True, returns the frequencies read from the instrument

Returns values as configured with mode

## property frequency

AC frequency (range depending on model), in Hertz

## property impedance

Measured data A and B, according to mode

# property mode

Select quantities to be measured:

- CPD: Parallel capacitance [F] and dissipation factor [number]
- CPQ: Parallel capacitance [F] and quality factor [number]
- CPG: Parallel capacitance [F] and parallel conductance [S]
- CPRP: Parallel capacitance [F] and parallel resistance [Ohm]
- CSD: Series capacitance [F] and dissipation factor [number]
- CSQ: Series capacitance [F] and quality factor [number]
- CSRS: Series capacitance [F] and series resistance [Ohm]
- LPD: Parallel inductance [H] and dissipation factor [number]
- LPQ: Parallel inductance [H] and quality factor [number]
- LPG: Parallel inductance [H] and parallel conductance [S]

- LPRP: Parallel inductance [H] and parallel resistance [Ohm]
- LSD: Series inductance [H] and dissipation factor [number]
- LSQ: Seriesinductance [H] and quality factor [number]
- LSRS: Series inductance [H] and series resistance [Ohm]
- RX: Resistance [Ohm] and reactance [Ohm]
- ZTD: Impedance, magnitude [Ohm] and phase [deg]
- ZTR: Impedance, magnitude [Ohm] and phase [rad]
- GB: Conductance [S] and susceptance [S]
- YTD: Admittance, magnitude [Ohm] and phase [deg]
- YTR: Admittance magnitude [Ohm] and phase [rad]

## property trigger\_source

# Select trigger source; accept the values:

- HOLD: manual
- · INT: internal
- BUS: external bus (GPIB/LAN/USB)
- EXT: external connector

# 7.9.5 Agilent E5062A Vector Network Analyzer

Bases: SCPIMixin, Instrument

Represents the Agilent E5062A Vector Network Analyzer

This VNA has 4 separate channels, each of which has its own sweep. The channels are stored in the channels dictionary. Channels can be enabled even if not displayed. Many trace-specific operations, including reading out data, happen at the channel level, but pertain to only the *active trace*.

Each channel can display up to 4 traces (controlled via the visible\_traces parameter). Each channel also has a display\_layout property that controls the layout of the traces in the channel. The traces are accessed via the traces dictionary (i.e. vna.channels[1].traces[1]).

The VNA supports multiple transfer formats. This API only supports the IEEE 64-bit floating point format. The VNA is configured for this format during initialization, and the data transfer format is not a publicly accessible property.

This class implements only a subset of the total E5062A functionality. The more significant missing functionality includes (TODO):

- editing the scale of the display
- · markers
- calibration
- power sweeps

```
# connect to the VNA and make a measurement consisting of 10 averages
vna = AgilentE5062A("TCPIP::192.168.2.233::INSTR")
ch = vna.channels[1]
                                 # use channel 1
                                   # use all 4 traces
ch.visible_traces = 4
for i, (tr, parameter) in enumerate(zip(
        ch.traces.values(),
         ['S11', 'S12', 'S21', 'S22'])):
    tr.parameter = parameter
ch.averages = 10
                                   # use 10x averaging
ch.averaging_enabled = True # turn averaging on
ch.trigger_continuous = False # turn off continuous triggering
vna.trigger_source = 'BUS'  # require remote trigger
vna.abort()
                                  # stop the current sweep
                                  # clear current averaging
ch.restart_averaging()
for _ in range(ch.averages):
    ch.trigger_initiate()  # arm channel 1 for single sweep
vna.trigger_single()  # send a trigger
vna.wait_for_complete()  # wait until the sweep is complete
# see `agilentE5062A.data` for an example of saving the measurement
```

ch 1

#### Channel

**VNAChannel** 

ch\_2

### Channel

**VNAChannel** 

ch\_3

### Channel

**VNAChannel** 

ch\_4

### Channel

**VNAChannel** 

abort()

Abort the current sweep (affects the trigger fsm, see page 80 of the programmer's manual for details)

### property display\_layout

Control the layout of the windows (channels) on the display (str). See the list of valid options below:

- D1
- D12
- D1\_2
- D112
- D1\_1\_2
- D123

- D1 2 3
- D12 33
- D11\_23
- D13\_23
- D12 13
- D1234
- D1\_2\_3\_4
- D12 34

In general, an occurance of a number denotes the associated channel being visible and an occurence of '\_' denotes a vertical split. Multiple occurences of the same number denote the channel having a larger window than the other channels. Refer to Figure 3-1 in the the programmer's manual for details.

Note that this splits windows for multiple *channels*, not traces. Basic S-param measurements most likely want to use only a single channel, but with multuple *traces* instead. See e.g. AgilentE5062A. channels[1].visible\_traces

# property output\_enabled

Control whether to turn on the RF stimulus (bool). The stimulus needs to be on to perform any measurement. Beware that the stimulus is on by default! (i.e. after reset())

### pop\_err()

Pop an error off the error queue. Returns a tuple containing the code and error description. An error code 0 indicates success.

The Error queue can be cleared using the standard SCPI agilentE5062A.clear() method.

# trigger()

Regardless of the trigger setting, generate a trigger (if the trigger has been initialized, i.e. is in the trigger wait state)

#### trigger\_bus()

If the trigger source is BUS and the VNA is waiting for a trigger (the trigger has been initialized), generate a trigger.

# trigger\_single()

like trigger(), but the *complete* SCPI property (synchronization bit) waits for the sweep to be complete (see *agilentE5062A.wait\_for\_complete()*).

### property trigger\_source

Control the trigger source (str). From the documentation:

- INT: Uses the internal trigger to generate continuous triggers automatically (default).
- EXT: Generates a trigger when the trigger signal is inputted externally via the Ext Trig connector or the handler interface.
- MAN: Generates a trigger when the key operation of [Trigger] Trigger is executed from the front panel.
- BUS: Generates a trigger when the \*TRG command is executed.

# wait\_for\_complete(attempt=1, max\_attempts=20)

Wait a potentially long time for the synchronization bit. This is useful in conjunction with agilentE5062A.trigger\_single() to wait for a single sweep to complete.

class pymeasure.instruments.agilent.agilentE5062A.VNAChannel(\*args, \*\*kwargs)

Bases: Channel

A measurement channel of the E5061A/E5062A

## property IF\_bandwidth

Control the IF bandwidth in Hz (int), from 10 Hz to 30 kHz. Default 30 kHz.

#### activate()

Activate (select) the current channel. Note that the channel must first be displayed (via AgilentE5062A.display\_layout) for it to be activatable.

### property attenuation

Control the stimulus attenuation in dB (positive int), from 0 to 40 in increments of 10. Default is 0 dB. The allowable stimulus power range is a 15 dB range: (attenuation - 5 dB, attenuation + 10 dB).

This requires the power range extension, and the command is ignored if the extension is not installed.

### property averages

Control how many averages to take, from 1-999 (int). Note that averaging\_enabled needs to be true for averaging to be enabled.

# property averaging\_enabled

Control whether to average the measurement data (bool).

#### property data

Measure the Formatted Data of the active trace.

Each trace consists of scan\_points plotted either vs. frequency or in something like a smith-chart configuration. This property does not access any frequency information. So for rectangular plots, this query returns only the y-values of the trace (no frequency information). For smith- and polar- plots, this two values per data point.

This function returns a tuple containing both a primary and a secondary data numpy array. The secondary data is all zeros for all trace formats that are not smith or polar. The implication for this is that the best way to save complex S-parameter data in one go is to use a Smith (Real/Imag) or Polar (Real/Imag) trace format, (SCOMplex and POLar, respectively).

The formatted data array is settable in the VNA but not implemented in this python API.

Frequency data is accessed via the frequencies property.

```
# build an s-parameter matrix by measuring the trace data,
# where s_matrix[i] = [[S11, S12], [S21, S22]] @ freqs[i]
freqs = ch.frequencies
s_matrix = np.empty((freqs.size, 4), dtype=np.complex64)
for i, tr in enumerate(ch.traces.values()):
    tr.activate()
    ch.trace_format = 'POL'
    re, im = ch.data
    s_matrix[:, i] = re + 1j * im
s_matrix = s_matrix.reshape(-1, 2, 2) # ready for use with e.g. skrf
```

### property display\_layout

Control the graph layout of the traces in the channel (str). Does not affect how many traces are active. See the list of valid options:

• D1

- D12
- D1 2
- D112
- D1\_1\_2
- D123
- D1 2 3
- D12\_33
- D11\_23
- D13 23
- D12 13
- D1234
- D1\_2\_3\_4
- D12\_34

In general, an occurance of a number denotes the associated trace having it's own subplot and an occurence of '\_' denotes a vertical split. Multiple occurences of the same number denote the trace having a larger window than the other traces. Refer to Figure 3-1 in the the programmer's manual for details. If a trace is active but it's associated number is not present in the display\_layout identifier, it is plotted in the first subplot.

### property frequencies

Measure the frequency in Hz associated with each data point of the active trace. Returns a numpy array.

# property power

Control the simulus power in dBm (float). The allowable range is influenced by the value of attenuation. (dynamic)

# restart\_averaging()

Clear averaging history.

# property scan\_points

Control the number of points used in a sweep (int). Valid range 2 - 1601.

### property start\_frequency

Control the start frequency in Hz (float).

### property stop\_frequency

Control the stop frequency in Hz (float).

# property sweep\_time

Control the sweep time in seconds (float). The allowable range varies on config and the set value is truncated. Note that sweep\_time\_auto\_enabled needs to be False for changes to this property to have an effect.

## property sweep\_time\_auto\_enabled

Control whether to automatically set the sweep time (bool). You probably want this on to always keep the sweep time to a minimum (given the range, IF BW, and sweep delay time).

### property sweep\_type

Control the type of the sweep (string).

Setting	Description
LIN	Linear
LOG	Logarithmic
SEGM	Segment
POW	Power

Defaults to linear. Note that the API for configuring segment type sweeps is not implememented in this class.

# property trace\_format

Control the data format of the *active trace* of the channel (str). Default is MLOGarithmic. From the programmer's manual:

Setting	Description
MLOG	Specifies the logarithmic magnitude format.
PHAS	Specifies the phase format.
GDEL	Specifies the group delay format.
SLIN	Specifies the Smith chart format (Lin/Phase).
SLOG	Specifies the Smith chart format (Log/Phase).
SCOM	Specifies the Smith chart format (Real/Imag).
SMIT	Specifies the Smith chart format $(R+jX)$ .
SADM	Specifies the Smith chart format (G+jB).
PLIN	Specifies the polar format (Lin).
PLOG	Specifies the polar format (Log).
POr	Specifies the polar format (Re/Im).
MLI	Specifies the linear magnitude format.
SWR	Specifies the SWR format.
REAL	Specifies the real format.
IMAG	Specifies the imaginary format.
UPH	Specifies the expanded phase format.
PPH	Specifies the positive phase format.

# property trigger\_continuous

Control whether the channel triggers continuously (bool). If not, after a sweep is complete, the global trigger enters the hold state, and will not respond to triggers.

# trigger\_initiate()

If the trigger is in the hold state (i.e. trigger\_continuous is off and the measurement has completed), re-initialize the trigger for a single measurement.

# property visible\_traces

Control the number of traces visible in the channel (int).

class pymeasure.instruments.agilent.agilentE5062A.VNATrace(parent, id)

Bases: Channel

A trace (of a channel) of the E5061A/E5062A

```
activate()
```

Select this trace.

# property parameter

Control the measurement parameter of the trace (str). Can be S11, S21, S12, or S22

# 7.9.6 Agilent 34410A Multimeter

Bases: SCPIUnknownMixin. Instrument

Represent the HP/Agilent/Keysight 34410A and related multimeters.

Implemented measurements: voltage\_dc, voltage\_ac, current\_dc, current\_ac, resistance, resistance\_4w

# property current\_ac

Get AC current, in Amps

### property current\_dc

Get DC current, in Amps

### property resistance

Get Resistance, in Ohms

### property resistance\_4w

Get Four-wires (remote sensing) resistance, in Ohms

## property voltage\_ac

Get AC voltage, in Volts

# property voltage\_dc

Get DC voltage, in Volts

# 7.9.7 HP/Agilent/Keysight 34450A Digital Multimeter

Bases: SCPIUnknownMixin, Instrument

Represent the HP/Agilent/Keysight 34450A and related multimeters.

```
dmm = Agilent34450A("USB0::...")
dmm.reset()
dmm.configure_voltage()
print(dmm.voltage)
dmm.shutdown()
```

#### beep()

Sounds a system beep.

# property capacitance

Get a capacitance measurement in Farads, based on the active *mode*.

#### property capacitance\_auto\_range

Control auto ranging for capacitance.

# property capacitance\_range

Control the capacitance range in Farads, which can take values 1E-9, 10E-9, 10E-9, 1E-6, 10E-6, 10E-6, 1E-3, 10E-3, as well as "MIN", "MAX", or "DEF" (1E-6). Auto-range is disabled when this property is set.

### configure\_capacitance(capacitance range='AUTO')

Configures the instrument to measure capacitance.

# **Parameters**

**capacitance\_range** – A capacitance in Farads to set the capacitance range, can be 1E-9, 10E-9, 10E-9, 1E-6, 10E-6, 10E-6, 1E-3, 10E-3, as well as "MIN", "MAX", "DEF" (1E-6), or "AUTO".

# configure\_continuity()

Configures the instrument to measure continuity.

### **configure\_current**(current\_range='AUTO', ac=False, resolution='DEF')

Configures the instrument to measure current.

#### **Parameters**

- current\_range A current in Amps to set the current range. DC values can be 100E-6, 1E-3, 10E-3, 100E-3, 1, 10, as well as "MIN", "MAX", "DEF" (100 mA), or "AUTO". AC values can be 10E-3, 100E-3, 1, 10, as well as "MIN", "MAX", "DEF" (100 mA), or "AUTO".
- ac False for DC current, and True for AC current
- **resolution** Desired resolution, can be 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

## configure\_diode()

Configures the instrument to measure diode voltage.

**configure\_frequency**(*measured\_from='voltage\_ac'*, *measured\_from\_range='AUTO'*, *aperture='DEF'*)

Configures the instrument to measure frequency.

# **Parameters**

- measured\_from "voltage ac" or "current ac"
- measured\_from\_range range of measured\_from. AC voltage can have ranges 100E-3, 1, 10, 100, 750, as well as "MIN", "MAX", "DEF" (10 V), or "AUTO". AC current can have ranges 10E-3, 100E-3, 1, 10, as well as "MIN", "MAX", "DEF" (100 mA), or "AUTO".
- aperture Aperture time in Seconds, can be 100 ms, 1 s, as well as "MIN", "MAX", or "DEF" (1 s).

## configure\_resistance(resistance\_range='AUTO', wires=2, resolution='DEF')

Configures the instrument to measure resistance.

# **Parameters**

- resistance\_range A resistance in Ohms to set the resistance range, can be 100, 1E3, 10E3, 10E3, 10E6, 10E6, 10E6, 10E6, as well as "MIN", "MAX", "DEF" (1E3), or "AUTO".
- wires Number of wires used for measurement, can be 2 or 4.

• **resolution** – Desired resolution, can be 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

# configure\_temperature()

Configures the instrument to measure temperature.

### **configure\_voltage**(voltage\_range='AUTO', ac=False, resolution='DEF')

Configures the instrument to measure voltage.

#### **Parameters**

- voltage\_range A voltage in Volts to set the voltage range. DC values can be 100E-3, 1, 10, 100, 1000, as well as "MIN", "MAX", "DEF" (10 V), or "AUTO". AC values can be 100E-3, 1, 10, 100, 750, as well as "MIN", "MAX", "DEF" (10 V), or "AUTO".
- ac False for DC voltage, True for AC voltage
- **resolution** Desired resolution, can be 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

# property continuity

Get a continuity measurement in Ohms, based on the active mode.

### property current

Get a DC current measurement in Amps, based on the active *mode*.

### property current\_ac

Get an AC current measurement in Amps, based on the active *mode*.

#### property current\_ac\_auto\_range

Control auto ranging for AC current.

# property current\_ac\_range

Control the AC current range in Amps, which can take values 10E-3, 100E-3, 1, 10, as well as "MIN", "MAX", or "DEF" (100 mA). Auto-range is disabled when this property is set.

# property current\_ac\_resolution

Control the resolution in the AC current readings, which can take values 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

#### property current\_auto\_range

Control auto ranging for DC current.

### property current\_range

Control the DC current range in Amps, which can take values 100E-6, 1E-3, 10E-3, 100E-3, 1, 10, as well as "MIN", "MAX", or "DEF" (100 mA). Auto-range is disabled when this property is set.

## property current\_resolution

Control the resolution in the DC current readings, which can take values 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", and "DEF" (3.00E-5).

### property diode

Get a diode measurement in Volts, based on the active *mode*.

### property frequency

Get a frequency measurement in Hz, based on the active mode.

# property frequency\_aperture

Control the frequency aperture in seconds, which sets the integration period and measurement speed. Takes values 100 ms, 1 s, as well as "MIN", "MAX", or "DEF" (1 s).

# property frequency\_current\_auto\_range

Control whether auto ranging for AC current in frequency measurements is enabled.

# property frequency\_current\_range

Control the current range in Amps for frequency on AC current measurements, which can take values 10E-3, 100E-3, 1, 10, as well as "MIN", "MAX", or "DEF" (100 mA). Auto-range is disabled when this property is set.

### property frequency\_voltage\_auto\_range

Control whether auto ranging for AC voltage in frequency measurements is enabled.

### property frequency\_voltage\_range

Control the voltage range in Volts for frequency on AC voltage measurements, which can take values 100E-3, 1, 10, 100, 750, as well as "MIN", "MAX", or "DEF" (10 V). Auto-range is disabled when this property is set.

### property mode

Control the measurement mode of the multimeter. Can be "current", "ac current", "voltage", "ac voltage", "resistance", "4w resistance", "current frequency", "voltage frequency", "continuity", "diode", "temperature", or "capacitance".

# property resistance

Get a resistance measurement in Ohms for 2-wire configuration, based on the active mode.

#### property resistance\_4w

Get a resistance measurement in Ohms for 4-wire configuration, based on the active mode.

### property resistance\_4w\_auto\_range

Control auto ranging for 4-wire resistance.

### property resistance\_4w\_range

Control the 4-wire resistance range in Ohms, which can take values 100, 1E3, 10E3, 10E3, 10E6, 10E6, 100E6, as well as "MIN", "MAX", or "DEF" (1E3). Auto-range is disabled when this property is set.

#### property resistance\_4w\_resolution

Control the resolution in the 4-wire resistance readings, which can take values 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

## property resistance\_auto\_range

Control auto ranging for 2-wire resistance.

### property resistance\_range

Control the 2-wire resistance range in Ohms, which can take values 100, 1E3, 10E3, 10E3, 10E6, 10E6, 100E6, as well as "MIN", "MAX", or "DEF" (1E3). Auto-range is disabled when this property is set.

### property resistance\_resolution

Control the resolution in the 2-wire resistance readings, which can take values 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

# property temperature

Get a temperature measurement in Celsius, based on the active *mode*.

### property voltage

Get a DC voltage measurement in Volts, based on the active *mode*.

# property voltage\_ac

Get an AC voltage measurement in Volts, based on the active mode.

### property voltage\_ac\_auto\_range

Control auto ranging for AC voltage.

# property voltage\_ac\_range

Control the AC voltage range in Volts, which can take values 100E-3, 1, 10, 100, 750, as well as "MIN", "MAX", or "DEF" (10 V). Auto-range is disabled when this property is set.

### property voltage\_ac\_resolution

Control the resolution in the AC voltage readings, which can take values 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

### property voltage\_auto\_range

Control auto ranging for DC voltage.

### property voltage\_range

Control the DC voltage range in Volts, which can take values 100E-3, 1, 10, 100, 1000, as well as "MIN", "MAX", or "DEF" (10 V). Auto-range is disabled when this property is set.

# property voltage\_resolution

Control the resolution in the DC voltage readings, which can take values 3.00E-5, 2.00E-5, 1.50E-6 (5 1/2 digits), as well as "MIN", "MAX", or "DEF" (1.50E-6).

# 7.9.8 Agilent 4155/4156 Semiconductor Parameter Analyzer

Bases: SCPIUnknownMixin, Instrument

Represents the Agilent 4155/4156 Semiconductor Parameter Analyzer and provides a high-level interface for taking current-voltage (I-V) measurements.

The JSON file is an ascii text configuration file that defines the settings of each channel on the instrument. The JSON file is used to configure the instrument using the convenience function <code>configure()</code> as shown in the example above. For example, the instrument setup for a bipolar transistor measurement is shown below.

```
{
       "SMU1": {
           "voltage_name" : "VC",
           "current_name" : "IC",
           "channel_function" : "VAR1",
           "channel_mode" : "V",
           "series_resistance" : "00HM"
       },
       "SMU2": {
           "voltage_name" : "VB",
           "current_name" : "IB",
           "channel_function" : "VAR2",
           "channel_mode" : "I",
           "series_resistance" : "00HM"
       },
       "SMU3": {
           "voltage_name" : "VE",
           "current_name" : "IE",
           "channel_function" : "CONS",
           "channel_mode" : "V",
           "constant_value" : 0,
           "compliance": 0.1
       },
        "SMU4": {
           "voltage_name" : "VS",
           "current_name" : "IS",
           "channel_function" : "CONS",
           "channel_mode" : "V",
           "constant_value" : 0,
           "compliance" : 0.1
       },
       "VAR1": {
           "start" : 1,
           "stop" : 2,
           "step" : 0.1,
           "spacing" : "LINEAR",
           "compliance" : 0.1
       },
       "VAR2": {
           "start" : 0,
           "step" : 10e-6,
           "points" : 3.
           "compliance" : 2
```

(continues on next page)

```
}
```

### property analyzer\_mode

Control the instrument operating mode.

• Values: SWEEP, SAMPLING

```
smu.analyzer_mode = "SWEEP"
```

## configure(config\_file)

Configure the channel setup and sweep using a JSON configuration file.

(JSON is the JavaScript Object Notation)

### **Parameters**

**config\_file** – JSON file to configure instrument channels.

```
instr.configure('config.json')
```

# property data\_variables

Get a string list of data variables for which measured data is available.

This looks for all the variables saved by the <code>save()</code> and <code>save\_var()</code> methods and returns it. This is useful for creation of dataframe headers.

### **Returns**

List

```
header = instr.data_variables
```

## property delay\_time

Control the measurement delay time in seconds, which can take the values from 0 to 65s in 0.1s steps.

```
instr.delay_time = 1 # delay time of 1-sec
```

# disable\_all()

Disable all channels in the instrument.

```
instr.disable_all()
```

## get\_data(path=None)

Get the measurement data from the instrument after completion.

If the measurement period is set to INF in the *measure()* method, then the measurement must be stopped using *stop()* before getting valid data.

# **Parameters**

**path** – Path for optional data export to CSV.

#### Returns

Pandas Dataframe

```
df = instr.get_data(path='./datafolder/data1.csv')
```

### property hold\_time

Control the measurement hold time in seconds, which can take the values from 0 to 655s in 1s steps.

```
instr.hold_time = 2 # hold time of 2-secs.
```

# property integration\_time

Control the integration time.

• Values: SHORT, MEDIUM, LONG

```
instr.integration_time = "MEDIUM"
```

## measure(period='INF', points=100)

Perform a single measurement and wait for completion in sweep mode.

In sampling mode, the measurement period and number of points can be specified.

#### **Parameters**

- **period** Period of sampling measurement from 6E-6 to 1E11 seconds. Default setting is INF.
- **points** Number of samples to be measured, from 1 to 10001. Default setting is 100.

### save(trace list)

Save the voltage or current in the instrument display list

#### **Parameters**

**trace\_list** – A list of channel variables whose measured data should be saved. A maximum of 8 variables are allowed. If only one variable is being saved, a string can be specified.

```
instr.save(['IC', 'IB', 'VC', 'VB']) #for list of variables
instr.save('IC') #for single variable
```

# save\_var(trace\_list)

Save the voltage or current in the instrument variable list.

This is useful if one or two more variables need to be saved in addition to the 8 variables allowed by save ().

# **Parameters**

**trace\_list** – A list of channel variables whose measured data should be saved. A maximum of 2 variables are allowed. If only one variable is being saved, a string can be specified.

```
instr.save_var(['VA', 'VB'])
```

### stop()

Stop the ongoing measurement.

```
instr.stop()
```

 $\textbf{class} \ \ \textbf{pymeasure.instruments.agilent.agilent4156.} \\ \textbf{AgilentMeasurementChannel} (\textit{parent}, id)$ 

Bases: Channel

Offers some common properties.

# property channel\_function

Control the channel function.

• Values: VAR1, VAR2, VARD or CONS.

### property channel\_mode

Control the channel mode.(dynamic)

### property disable

Set the settings of this channel to default value.

```
[instr.smu1.disable
```

## reset\_settings()

Reset the settings of this channel to default value.

```
instr.smu1.reset_settings()
```

## property voltage\_name

Control the voltage name of the channel.

If input is greater than 6 characters long or starts with a number, the name is autocorrected and prepended with 'a'. Event is logged.

```
instr.smu1.voltage_name = "Vbase"
```

# class pymeasure.instruments.agilent.agilent4156.SMU(parent, id, \*\*kwargs)

Bases: AgilentMeasurementChannel

SMU of Agilent 4155/4156 Semiconductor Parameter Analyzer

# property compliance

Control the *constant* compliance value of SMU<n>.

If the SMU channel is setup as a variable (VAR1, VAR2, VARD) then compliance limits are set by the variable definition.

• Value: Voltage in (-200V, 200V) and current in (-1A, 1A) based on channel\_mode().

```
instr.smu1.compliance = 0.1
```

### property constant\_value

Control the constant source value of SMU<n>.

You use this command only if channel\_function() is CONS and also channel\_mode() should not be COMM.

#### **Parameters**

**const\_value** – Voltage in (-200V, 200V) and current in (-1A, 1A). Voltage or current depends on if channel\_mode() is set to V or I.

```
[instr.smu1.constant_value = 1
```

# property current\_name

Control the current name of the channel.

If input is greater than 6 characters long or starts with a number, the name is autocorrected and prepended with 'a'. Event is logged.

```
instr.smu1.voltage_name = "Vbase"
```

### property series\_resistance

Control the series resistance of SMU<n>.

Values: OOHM, 10KOHM, 100KOHM, or 1MOHM

```
instr.smu1.series_resistance = "10KOHM"
```

class pymeasure.instruments.agilent.agilent4156.VAR1(parent, \*\*kwargs)

Bases: VARX

Class to handle all the specific definitions needed for VAR1. Most common methods are inherited from base class

# property spacing

Control the sweep type of VAR1.

• Values: LINEAR, LOG10, LOG25, LOG50.

class pymeasure.instruments.agilent.agilent4156.VAR2(adapter, \*\*kwargs)

Bases: VARX

Class to handle all the specific definitions needed for VAR2. Common methods are imported from base class.

### property points

Control the number of sweep steps of VAR2. You use this command only if there is an SMU or VSU whose function (FCTN) is VAR2.

```
[instr.var2.points = 10
```

class pymeasure.instruments.agilent.agilent4156.VARD(parent, id='VARD', \*\*kwargs)

Bases: Channel

Class to handle all the definitions needed for VARD. VARD is always defined in relation to VAR1.

# property channel\_mode

Control the channel mode.

### property compliance

Control the sweep COMPLIANCE value of VARD.

```
instr.vard.compliance = 0.1
```

### property offset

Control the OFFSET value of VARD. For each step of sweep, the output values of VAR1' are determined by the following equation: VARD = VAR1 X RATio + OFFSet You use this command only if there is an SMU or VSU whose function is VARD.

```
instr.vard.offset = 1
```

# property ratio

Control the RATIO of VAR1'. For each step of sweep, the output values of VAR1' are determined by the following equation: VAR1' = VAR1 \* RATio + OFFSet You use this command only if there is an SMU or VSU whose function (FCTN) is VAR1'.

```
instr.vard.ratio = 1
```

```
class pymeasure.instruments.agilent.agilent4156.VARX(parent, id, **kwargs)
```

Bases: Channel

Base class to define sweep variable settings.

### property channel\_mode

Control the channel mode.

# property compliance

Control the sweep COMPLIANCE value.

```
instr.var1.compliance = 0.1
```

# property start

Control the sweep START value.

```
instr.var1.start = 0
```

# property step

Control the sweep STEP value.

```
instr.var1.step = 0.1
```

# property stop

Control the sweep STOP value.

```
[instr.var1.stop = 3
```

class pymeasure.instruments.agilent.agilent4156.VMU(parent, id, \*\*\*kwargs)

Bases: AgilentMeasurementChannel

VMU of Agilent 4155/4156 Semiconductor Parameter Analyzer

class pymeasure.instruments.agilent.agilent4156.VSU(parent, id, \*\*kwargs)

Bases: AgilentMeasurementChannel

VSU of Agilent 4155/4156 Semiconductor Parameter Analyzer

# property constant\_value

Control the constant source value of VSU<n>.

```
instr.vsu1.constant_value = 0
```

# 7.9.9 Agilent 4294A Precision Impedance Analyzer

class pymeasure.instruments.agilent.Agilent4294A(adapter, name='Agilent 4294A Precision Impedance Analyzer', read\_termination=\n', write\_termination=\n', timeout=5000, \*\*kwargs)

Bases: SCPIMixin, Instrument

Represents the Agilent 4294A Precision Impedance Analyzer

### property active\_trace

Control the active trace

### get\_data(path=None)

Get the measurement data from the instrument after completion.

#### **Parameters**

**path** – Path for optional data export to CSV.

#### Returns

Pandas Dataframe

# property measurement\_type

Control the measurement type. See MEASUREMENT\_TYPES

# property num\_points

Control the number of points measured at each sweep

```
save_graphics(path=")
```

Save graphics on the screen to a file on the local computer. Adapted from: https://www.keysight.com/se/en/lib/software-detail/programming-examples/4294a-data-transfer-program-excel-vba-1645196.html

### property start\_frequency

Control the start frequency in Hz

# property stop\_frequency

Control the stop frequency in Hz

# property title

Control the title of the active trace

# 7.9.10 Agilent 33220A Arbitrary Waveform Generator

Bases: SCPIUnknownMixin, Instrument

Represents the Agilent 33220A Arbitrary Waveform Generator.

```
# Default channel for the Agilent 33220A
wfg = Agilent33220A("GPIB::10")
wfg.shape = "SINUSOID"
                                 # Sets a sine waveform
wfg.frequency = 4.7e3
                                   # Sets the frequency to 4.7 kHz
wfg.amplitude = 1
                                   # Set amplitude of 1 V
wfg.offset = 0
                                    # Set the amplitude to 0 V
wfg.burst_mode # Enable burst mode # A burst mode
                                    # A burst will consist of 10 cycles
wig.burst_ncycles = 10  # A burst will consist of 10 cycles
wfg.burst_mode = "TRIGGERED"  # A burst will be applied on a trigger
wfg.trigger_source = "BUS"  # A burst will be triggered on TRG*
wfg.output = True
                                     # Enable output of waveform generator
                                     # Trigger a burst
wfg.trigger()
                                    # Wait until the triggering is finished
wfg.wait_for_trigger()
                                     # "beep"
wfg.beep()
print(wfg.check_errors())
                                     # Get the error queue
```

### property amplitude

Control the amplitude of the output waveform. Depending on amplitude unit, the unit is Volt (peak-to-peak), Volt (RMS) or dBm. The limits depend on the configured output termination (50 Ohm to High Impedance changes by a factor of 2) and the offset. Supplying out-of-bound values may silently be clipped by the device (float, in V or dBm)

# property amplitude\_unit

Control the units of the amplitude (string, strict from VPP, VRMS, DBM).

## beep()

Causes a system beep.

### property beeper\_state

Control the state of the beeper (bool).

### property burst\_mode

Control the burst mode (string, strict from TRIG, TRIGGERED, GAT, GATED).

## property burst\_ncycles

Control the number of cycles to be output when a burst is triggered (int, strict from 1 to 50000).

### property burst\_state

Control whether the burst mode is on (bool).

### property frequency

Control the frequency of the output waveform. Depending on the output shape, the supported frequency range changes. Supplying out-of-bound values may silently be clipped by the device (float, in Hz)

# property offset

Control the voltage offset of the output waveform. This is limited by the amplitude and output termination. Supplying out-of-bound values may silently be clipped by the device. (float, in V)

# property output

Control the output of the function generator (bool).

### property pulse\_dutycycle

Control the duty cycle of a pulse waveform function in percent (float, strict from 0 to 100).

# property pulse\_hold

Control if either the pulse width or the duty cycle is retained when changing the period or frequency of the waveform (string, strict from WIDT, WIDTH, DCYC, DCYCLE).

### property pulse\_period

Control the period of a pulse waveform function in seconds (float, strict from 200e-9 to 2e3). The period overwrites the frequency for all waveforms. If the period is shorter than the pulse width + the edge time, the edge time and pulse width will be adjusted accordingly.

# property pulse\_transition

Control the edge time in seconds for both the rising and falling edges. It is defined as the time between 0.1 and 0.9 of the threshold. Valid values are between 5 ns to 100 ns. The transition time has to be smaller than 0.625 \* the pulse width.

### property pulse\_width

Control the width of a pulse waveform function in seconds (float, strict from 20e-9 to 2e3).

# property ramp\_symmetry

Control the symmetry percentage for the ramp waveform (float, strict from 0 to 100).

#### property remote\_local\_state

Set the remote/local state of the function generator (string, strict from LOC, LOCAL, REM, REMOTE, RWL, RWLOCK).

### property shape

Control the output waveform (string, strict from SINUSOID, SIN, SQUARE, SQU, RAMP, PULSE, PULS, NOISE, NOIS, DC, USER).

### property square\_dutycycle

Control the duty cycle of a square waveform function (float, strict from 20 to 80).

### trigger()

Send a trigger signal to the function generator.

### property trigger\_source

Control the trigger source (string, strict from IMM, IMMEDIATE, EXT, EXTERNAL, BUS).

## property trigger\_state

Control whether the output is triggered (bool).

### property voltage\_high

Control the upper voltage of the output waveform. The limits depend on the output termination. Supplying out-of-bound values may silently be clipped by the device. (float, in V)

## property voltage\_low

Control the lower voltage of the output waveform. The limits depend on the output termination. Supplying out-of-bound values may silently be clipped by the device. (float, in V)

```
wait_for_trigger(timeout=3600, should_stop=<function Agilent33220A.<lambda>>)
```

Wait until the triggering has finished or timeout is reached.

### **Parameters**

- **timeout** The maximum time the waiting is allowed to take. If timeout is exceeded, a TimeoutError is raised. If timeout is set to zero, no timeout will be used.
- **should\_stop** Optional function (returning a bool) to allow the waiting to be stopped before its end.

# 7.9.11 Agilent 33500 Function/Arbitrary Waveform Generator Family

**class** pymeasure.instruments.agilent.**Agilent33500**(adapter, name='Agilent 33500 Function/Arbitrary Waveform generator family', \*\*kwargs)

```
Bases: SCPIMixin, Instrument
```

Represents the Agilent 33500 Function/Arbitrary Waveform Generator family.

Individual devices are represented by subclasses. User can specify a channel to control, if no channel specified, a default channel is picked based on the device e.g. For Agilent33500B the default channel is channel 1. See reference manual for your device

(continued from previous page)

```
→to sine
                                          # Sets default channel output frequency to.
generator.frequency = 1e3
\hookrightarrow 1 kHz
generator.channels[1].frequency = 1e3
                                                # Sets channel 1 output frequency to
\rightarrow 1 \text{ kHz}
                                               # Sets channel 2 output amplitude to...
generator.channels[2].amplitude = 5
→5 Vpp
generator.channels[2].output = 'on'
                                                # Enables channel 2 output
                                                # Set channel 1 shape to arbitrary
generator.channels[1].shape = 'ARB'
                                                # Set channel 1 sample rate to 1MSa/s
generator.channels[1].arb_srate = 1e6
generator.channels[1].data_volatile_clear() # Clear channel 1 volatile internal_
\hookrightarrow memory
generator.channels[1].data_arb(
                                               # Send data of arbitrary waveform to_
→channel 1
    'test'.
    range(-10000, 10000, +20),
                                          # In this case a simple ramp
    data_format='DAC'
                                          # Data format is set to 'DAC'
                                                # Select the transmitted waveform 'test
generator.channels[1].arb_file = 'test'
```

## ch\_1

#### Channel

Agilent33500Channel

ch\_2

#### Channel

Agilent33500Channel

## property amplitude

Control the voltage amplitude in Volts (float).(dynamic)

#### property amplitude\_unit

Control the amplitude units (string, strictly 'VPP', 'VRMS', or 'DBM').(dynamic)

#### property arb\_advance

Control how the device advances from data point to data point (str). Can be set to 'TRIG<GER>' or 'SRAT<E>' (default).

## property arb\_file

Control the arbitrary signal to use from the volatile memory of the device.(dynamic)

#### property arb\_filter

Control the filter setting for arbitrary signals (str). Can be set to 'NORM<AL>', 'STEP' and 'OFF'.

#### property arb\_srate

Control the sample rate of the currently selected arbitrary signal in Sa/s (float). Valid values range from 1  $\mu$ Sa/s to 250 MSa/s (maximum range, device-dependent).

## beep()

Causes a system beep.

#### property burst\_mode

Control the burst mode type (str, strictly 'TRIG<GERED>' or 'GAT<ED>').(dynamic)

## property burst\_ncycles

Control the number of cycles to be output when a burst is triggered (int).(dynamic)

## property burst\_period

Control the period of subsequent bursts in seconds (float).(dynamic)

#### property burst\_state

Control the burst mode state (bool).(dynamic)

## clear\_display()

Removes a text message from the display.

# data\_arb(arb\_name, data\_points, data\_format='DAC')

Uploads an arbitrary trace into the volatile memory of the device.

The data\_points can be given as: comma separated 16 bit DAC values (ranging from -32767 to +32767), as comma separated floating point values (ranging from -1.0 to +1.0) or as a binary data stream. Check the manual for more information. The storage depends on the device type and ranges from 8 Sa to 16 MSa (maximum).

#### **Parameters**

- arb\_name The name of the trace in the volatile memory. This is used to access the trace.
- data\_points Individual points of the trace. The format depends on the format parameter. format = 'DAC' (default): Accepts list of integer values ranging from -32767 to +32767. Minimum of 8 a maximum of 65536 points. format = 'float': Accepts list of floating point values ranging from -1.0 to +1.0. Minimum of 8 a maximum of 65536 points. format = 'binary': Accepts a binary stream of 8 bit data.
- data\_format Defines the format of data\_points. Can be 'DAC' (default), 'float' or 'binary'. See documentation on parameter data\_points above.

#### data\_volatile\_clear()

Clear all arbitrary signals from volatile memory.

This should be done if the same name is used continuously to load different arbitrary signals into the memory, since an error will occur if a trace is loaded which already exists in the memory.

#### property display

Set text to be displayed on the front panel of the device (string).

#### property ext\_trig\_out

Control whether the trigger out signal is active (bool).

#### property frequency

Control the waveform frequency in Hz (float). Depends on the specified shape.(dynamic)

# property offset

Control the voltage offset in Volts (float).(dynamic)

#### property output

Control the output state (bool).(dynamic)

# property output\_load

Control the expected load resistance in Ohms (str or float). The output impedance is always 50 Ohm, this setting can be used to correct the displayed voltage for loads unmatched to 50 Ohm.(dynamic)

#### property phase

Control the waveform phase in degrees (float, from -360 to 360).

# phase\_sync()

Synchronize the phase of all channels.

# property pulse\_dutycycle

Control the pulse duty cycle in percent (float, from 0 to 100).(dynamic)

#### property pulse\_hold

Control whether pulse width or duty cycle is maintained when the period or frequency of the waveform is changed (str).(dynamic)

# property pulse\_period

Control the pulse period in seconds (float). Overwrites frequency. If the period is shorter than the pulse width + the edge time, the edge time and pulse width are adjusted.(dynamic)

## property pulse\_transition

Control the pulse edge time (both rising and falling) in seconds (float).(dynamic)

#### property pulse\_width

Control the pulse width in seconds (float).(dynamic)

#### property ramp\_symmetry

Control the ramp waveform symmetry in percent (float, from 0 to 100).(dynamic)

#### property shape

Control the output waveform shape (str).(dynamic)

## property square\_dutycycle

Control the square wave duty cycle in percent (float).(dynamic)

## trigger()

Send a trigger signal to the function generator.

#### property trigger\_source

Control the trigger source (str).

# property voltage\_high

Control the upper voltage level in Volts (float).(dynamic)

#### property voltage\_low

Control the lower voltage level in Volts (float).(dynamic)

## wait\_for\_trigger(timeout=3600, should\_stop=<function Agilent33500.<lambda>>)

Wait until the triggering has finished or timeout is reached.

## **Parameters**

- **timeout** The maximum time the waiting is allowed to take. If timeout is exceeded, a TimeoutError is raised. If timeout is set to zero, no timeout will be used.
- **should\_stop** Optional function (returning a bool) to allow the waiting to be stopped before its end.

# 7.9.12 Agilent 33521A Function/Arbitrary Waveform Generator

## class pymeasure.instruments.agilent.Agilent33521A(adapter, \*\*kwargs)

Bases: Agilent33500

Represents the Agilent 33521A Function/Arbitrary Waveform Generator.

This documentation page shows only methods different from the parent class Agilent 33500.

#### $ch_1$

#### Channel

Agilent33500Channel

#### $ch_2$

#### Channel

Agilent33500Channel

# property arb\_srate

An floating point property that sets the sample rate of the currently selected arbitrary signal. Valid values are 1  $\mu$ Sa/s to 250 MSa/s. This can be set.

## property frequency

A floating point property that controls the frequency of the output waveform in Hz, from 1 uHz to 30 MHz, depending on the specified function. Can be set.

## class pymeasure.instruments.agilent.agilent33500.Agilent33500Channel(parent, id)

Bases: Channel

Implementation of a base Agilent 33500 channel

## property amplitude

Control the voltage amplitude in Volts (float).(dynamic)

#### property amplitude\_unit

Control the amplitude units (string, strictly 'VPP', 'VRMS', or 'DBM').(dynamic)

## property arb\_advance

Control how the device advances from data point to data point (str). Can be set to 'TRIG<GER>' or 'SRAT<E>' (default).

#### property arb\_file

Control the arbitrary signal to use from the volatile memory of the device.(dynamic)

# property arb\_filter

Control the filter setting for arbitrary signals (str). Can be set to 'NORM<AL>', 'STEP' and 'OFF'.

# property arb\_srate

Control the sample rate of the currently selected arbitrary signal in Sa/s (float). Valid values range from 1  $\mu$ Sa/s to 250 MSa/s (maximum range, device-dependent).

## property burst\_mode

Control the burst mode type (str, strictly 'TRIG<GERED>' or 'GAT<ED>').(dynamic)

# property burst\_ncycles

Control the number of cycles to be output when a burst is triggered (int).(dynamic)

#### property burst\_period

Control the period of subsequent bursts in seconds (float).(dynamic)

# property burst\_state

Control the burst mode state (bool).(dynamic)

#### data\_arb(arb\_name, data\_points, data\_format='DAC')

Uploads an arbitrary trace into the volatile memory of the device for a given channel.

The data\_points can be given as: comma separated 16 bit DAC values (ranging from -32767 to +32767), as comma separated floating point values (ranging from -1.0 to +1.0), or as a binary data stream. Check the manual for more information. The storage depends on the device type and ranges from 8 Sa to 16 MSa (maximum).

## **Parameters**

- arb\_name The name of the trace in the volatile memory. This is used to access the trace.
- data\_points Individual points of the trace. The format depends on the format parameter.

format = 'DAC' (default): Accepts list of integer values ranging from -32767 to +32767. Minimum of 8 a maximum of 65536 points.

format = 'float': Accepts list of floating point values ranging from -1.0 to +1.0. Minimum of 8 a maximum of 65536 points.

format = 'binary': Accepts a binary stream of 8 bit data.

• data\_format – Defines the format of data\_points. Can be 'DAC' (default), 'float' or 'binary'. See documentation on parameter data\_points above.

## data\_volatile\_clear()

Clear all arbitrary signals from volatile memory for a given channel.

This should be done if the same name is used continuously to load different arbitrary signals into the memory, since an error will occur if a trace is loaded which already exists in memory.

#### property frequency

Control the waveform frequency in Hz (float). Depends on the specified shape.(dynamic)

# property offset

Control the voltage offset in Volts (float).(dynamic)

## property output

Control the output state (bool).(dynamic)

## property output\_load

Control the expected load resistance in Ohms (str or float). The output impedance is always 50 Ohm, this setting can be used to correct the displayed voltage for loads unmatched to 50 Ohm.(dynamic)

# property phase

Control the waveform phase in degrees (float, from -360 to 360).

#### property pulse\_dutycycle

Control the pulse duty cycle in percent (float, from 0 to 100).(dynamic)

## property pulse\_hold

Control whether pulse width or duty cycle is maintained when the period or frequency of the waveform is changed (str).(dynamic)

## property pulse\_period

Control the pulse period in seconds (float). Overwrites frequency. If the period is shorter than the pulse width + the edge time, the edge time and pulse width are adjusted.(dynamic)

## property pulse\_transition

Control the pulse edge time (both rising and falling) in seconds (float).(dynamic)

## property pulse\_width

Control the pulse width in seconds (float).(dynamic)

#### property ramp\_symmetry

Control the ramp waveform symmetry in percent (float, from 0 to 100).(dynamic)

# property shape

Control the output waveform shape (str).(dynamic)

#### property square\_dutycycle

Control the square wave duty cycle in percent (float).(dynamic)

#### property voltage\_high

Control the upper voltage level in Volts (float).(dynamic)

## property voltage\_low

Control the lower voltage level in Volts (float).(dynamic)

# 7.9.13 Agilent B1500 Semiconductor Parameter Analyzer

# **Contents**

- Agilent B1500 Semiconductor Parameter Analyzer
  - General Information
    - \* Command Translation
  - Examples
    - \* Initialization of the Instrument
    - \* IV measurement with 4 SMUs
    - \* Sampling measurement with 4 SMUs
  - Main Classes
  - Supporting Classes
    - \* Enumerations

# **General Information**

This instrument driver does not support all configuration options of the B1500 mainframe yet. So far, it is possible to interface multiple SMU modules and source/measure currents and voltages, perform sampling and staircase sweep measurements. The implementation of further measurement functionalities is highly encouraged. Meanwhile the model is managed by Keysight, see the corresponding "Programming Guide" for details on the control methods and their parameters

# **Command Translation**

Alphabetical list of implemented B1500 commands and their corresponding method/attribute names in this instrument driver.

Command	Property/Method			
AAD	SMU.adc_type			
AB	abort()			
AIT	adc_setup()			
AV	adc_averaging()			
AZ	adc_auto_zero			
ВС	clear_buffer()			
CL	SMU.disable()			
CM	auto_calibration			
CMM	SMU.meas_op_mode			
CN	SMU.enable()			
DI	SMU.force() mode: 'CURRENT'			
DV	SMU.force() mode: 'VOLTAGE'			
DZ	<pre>force_gnd(), SMU.force_gnd()</pre>			
ERRX?	check_errors()			
FL	SMU.filter			
FMT	data_format()			
*IDN?	id			
*LRN?	query_learn(), multiple methods to read/format settings directly			
MI	<pre>SMU.sampling_source() mode: 'CURRENT'</pre>			
ML	sampling_mode			
MM	<pre>meas_mode()</pre>			
MSC	<pre>sampling_auto_abort()</pre>			
MT	<pre>sampling_timing()</pre>			
MV	<pre>SMU.sampling_source() mode: 'VOLTAGE'</pre>			
*0PC?	check_idle()			
PA	pause()			
PAD	parallel_meas			
RI	meas_range_current			
RM	SMU.meas_range_current_auto()			
*RST	reset()			
RV	meas_range_voltage			
SSR	series_resistor			
TSC	time_stamp			
TSR	clear_timer()			
UNT?	query_modules()			
WAT	wait_time()			
WI	SMU.staircase_sweep_source() mode: 'CURRENT'			

continues on next page

Table 1 – continued from previous page

Command	Property/Method
WM	<pre>sweep_auto_abort()</pre>
WSI	<pre>SMU.synchronous_sweep_source() mode: 'CURRENT'</pre>
WSV	<pre>SMU.synchronous_sweep_source() mode: 'VOLTAGE'</pre>
WT	<pre>sweep_timing()</pre>
WV	<pre>SMU.staircase_sweep_source() mode: 'VOLTAGE'</pre>
XE	send_trigger()

# **Examples**

## Initialization of the Instrument

```
from pymeasure.instruments.agilent import AgilentB1500

# explicitly define r/w terminations; set sufficiently large timeout in milliseconds or None.
b1500=AgilentB1500("GPIB0::17::INSTR", read_termination='\r\n', write_termination='\r\n', write_t
```

## IV measurement with 4 SMUs

```
# choose measurement mode
b1500.meas_mode('STAIRCASE_SWEEP', *b1500.smu_references) #order in smu_references_
→determines order of measurement
# settings for individual SMUs
for smu in b1500.smu_references:
    smu.enable() #enable SMU
    smu.adc_type = 'HRADC' #set ADC to high-resoultion ADC
    smu.meas_range_current = '1 nA'
    smu.meas_op_mode = 'COMPLIANCE_SIDE' # other choices: Current, Voltage, FORCE_SIDE,_
→ COMPLIANCE_AND_FORCE_SIDE
# General Instrument Settings
# b1500.adc_averaging = 1
# b1500.adc_auto_zero = True
b1500.adc_setup('HRADC','AUTO',6)
#b1500.adc_setup('HRADC', 'PLC', 1)
#Sweep Settings
b1500.sweep_timing(0,5,step_delay=0.1) #hold,delay
b1500.sweep_auto_abort(False,post='STOP') #disable auto abort, set post measurement_
→output condition to stop value of sweep
```

(continues on next page)

(continued from previous page)

```
# Sweep Source
nop = 11
b1500.smu1.staircase_sweep_source('VOLTAGE','LINEAR_DOUBLE','Auto Ranging',0,1,nop,0.
→001) #type, mode, range, start, stop, steps, compliance
# Synchronous Sweep Source
b1500.smu2.synchronous_sweep_source('VOLTAGE','Auto Ranging',0,1,0.001) #type, range, ___
⇔start, stop, comp
# Constant Output (could also be done using synchronous sweep source with start=stop, ...
→but then the output is not ramped up)
b1500.smu3.ramp_source('VOLTAGE','Auto Ranging',-1,stepsize=0.1,pause=20e-3) #output_
→starts immediately! (compared to sweeps)
b1500.smu4.ramp_source('VOLTAGE','Auto Ranging',0,stepsize=0.1,pause=20e-3)
#Start Measurement
b1500.check_errors()
b1500.clear_buffer()
b1500.clear_timer()
b1500.send_trigger()
# read measurement data all at once
b1500.check_idle() #wait until measurement is finished
data = b1500.read_data(2*nop) #Factor 2 because of double sweep
#alternatively: read measurement data live
meas = []
for i in range(nop*2):
   read_data = b1500.read_channels(4+1) # 4 measurement channels, 1 sweep source_
→ (returned due to mode=1 of data_format)
    # process live data for plotting etc.
    # data format for every channel (status code, channel name e.g. 'SMU1', data name e.g
→ 'Current Measurement (A)', value)
   meas.append(read_data)
#sweep constant sources back to OV
b1500.smu3.ramp_source('VOLTAGE','Auto Ranging',0,stepsize=0.1,pause=20e-3)
b1500.smu4.ramp_source('VOLTAGE','Auto Ranging',0,stepsize=0.1,pause=20e-3)
```

## Sampling measurement with 4 SMUs

(continued from previous page)

```
→ COMPLIANCE_AND_FORCE_SIDE
b1500.sampling_mode = 'LINEAR'
# b1500.adc_averaging = 1
# b1500.adc_auto_zero = True
b1500.adc_setup('HSADC','AUTO',1)
#b1500.adc_setup('HSADC', 'PLC', 1)
nop=11
b1500.sampling_timing(2,0.005,nop) #MT: bias hold time, sampling interval, number of
→points
b1500.sampling_auto_abort(False,post='BIAS') #MSC: BASE/BIAS
b1500.time_stamp = True
# Sources
b1500.smu1.sampling_source('VOLTAGE','Auto Ranging',0,1,0.001) #MV/MI: type, range, base,
→ bias, compliance
b1500.smu2.sampling_source('VOLTAGE','Auto Ranging',0,1,0.001)
b1500.smu3.ramp_source('VOLTAGE','Auto Ranging',-1,stepsize=0.1,pause=20e-3) #output_
→starts immediately! (compared to sweeps)
b1500.smu4.ramp_source('VOLTAGE','Auto Ranging',-1,stepsize=0.1,pause=20e-3)
#Start Measurement
b1500.check_errors()
b1500.clear_buffer()
b1500.clear_timer()
b1500.send_trigger()
meas=[]
for i in range(nop):
   read_data = b1500.read_channels(1+2*number_of_channels) #Sampling Index + (time_
→stamp + measurement value) * number of channels
    # process live data for plotting etc.
    # data format for every channel (status code, channel name e.g. 'SMU1', data name e.g
→ 'Current Measurement (A)', value)
   meas.append(read_data)
#sweep constant sources back to OV
b1500.smu3.ramp_source('VOLTAGE','Auto Ranging',0,stepsize=0.1,pause=20e-3)
b1500.smu4.ramp_source('VOLTAGE','Auto Ranging',0,stepsize=0.1,pause=20e-3)
```

## **Main Classes**

Classes to communicate with the instrument:

- AgilentB1500: Main instrument class
- SMU: Instantiated by main instrument class for every SMU

All *query* commands return a human readable dict of settings. These are intended for debugging/logging/file headers, not for passing to the accompanying setting commands.

```
Bases: SCPIUnknownMixin, Instrument
```

Represents the Agilent B1500 Semiconductor Parameter Analyzer and provides a high-level interface for taking different kinds of measurements.

# property smu\_references

Returns all SMU instances.

#### property smu\_names

Returns all SMU names.

# query\_learn(query\_type)

Queries settings from the instrument (\*LRN?). Returns dict of settings.

#### **Parameters**

```
query_type (int or str) – Query type (number according to manual)
```

## query\_learn\_header(query\_type, \*\*kwargs)

Queries settings from the instrument (\*LRN?). Returns dict of settings in human readable format for debugging or file headers. For optional arguments check the underlying definition of *QueryLearn*. *query\_learn\_header()*.

#### **Parameters**

```
query_type (int or str) – Query type (number according to manual)
```

#### reset()

Resets the instrument to default settings (\*RST)

## query\_modules()

Queries module models from the instrument. Returns dictionary of channel and module type.

#### Returns

Channel: Module Type

## Return type

dict

#### initialize\_smu(channel, smu\_type, name)

Initializes SMU instance by calling SMU.

# **Parameters**

- channel (int) SMU channel
- **smu\_type** (*str*) SMU type, e.g. 'HRSMU'
- name (str) SMU name for pymeasure (data output etc.)

## Returns

SMU instance

# **Return type**

**SMU** 

## initialize\_all\_smus()

Initialize all SMUs by querying available modules and creating a SMU class instance for each. SMUs are accessible via attributes .smu1 etc.

```
pause(pause_seconds)
```

Pauses Command Execution for given time in seconds (PA)

```
Parameters
             pause_seconds (int) - Seconds to pause
abort()
     Aborts the present operation but channels may still output current/voltage (AB)
force_gnd()
     Force 0V on all channels immediately. Current Settings can be restored with RZ. (DZ)
check_errors()
     Check for errors (ERRX?)
check_idle()
     Check if instrument is idle (*0PC?)
clear_buffer()
     Clear output data buffer (BC)
clear_timer()
     Clear timer count (TSR)
send_trigger()
     Send trigger to start measurement (except High Speed Spot) (XE)
property auto_calibration
     Enable/Disable SMU auto-calibration every 30 minutes. (CM)
             bool
data_format(output_format, mode=0)
     Specifies data output format. Check Documentation for parameters. Should be called once per session to
     set the data format for interpreting the measurement values read from the instrument. (FMT)
     Currently implemented are format 1, 11, and 21.
         Parameters
             • output_format (str) - Output format string, e.g. FMT21
             • mode (int, optional) – Data output mode, defaults to 0 (only measurement data is re-
               turned)
property parallel_meas
     Enable/Disable parallel measurements.
         Effective for SMUs using HSADC and measurement modes 1,2,10,18. (PAD)
         Type
             bool
query_meas_settings()
     Read settings for TM, AV, CM, FMT and MM commands (31) from the instrument.
query_meas_mode()
```

Read settings for MM command (part of 31) from the instrument.

```
meas_mode(mode, *args)
```

Set Measurement mode of channels. Measurements will be taken in the same order as the SMU references are passed. (MM)

#### **Parameters**

- mode (MeasMode) Measurement mode
  - Spot
  - Staircase Sweep
  - Sampling
- args (SMU) SMU references

# query\_adc\_setup()

Read ADC settings (55, 56) from the instrument.

```
adc_setup(adc_type, mode, N=")
```

Set up operation mode and parameters of ADC for each ADC type. (AIT) Defaults:

- HSADC: Auto N=1, Manual N=1, PLC N=1, Time N=0.000002(s)
- HRADC: Auto N=6, Manual N=3, PLC N=1

#### **Parameters**

- adc\_type (ADCType) ADC type
- mode (ADCMode) ADC mode
- N (str, optional) additional parameter, check documentation, defaults to ''

# adc\_averaging(number, mode='Auto')

Set number of averaging samples of the HSADC. (AV)

Defaults: N=1, Auto

#### **Parameters**

- **number** (int) Number of averages
- mode (AutoManual, optional) Mode ('Auto', 'Manual'), defaults to 'Auto'

# property adc\_auto\_zero

Enable/Disable ADC zero function. Halves the integration time, if off. (AZ)

#### **Type**

bool

# property time\_stamp

Enable/Disable Time Stamp function. (TSC)

## **Type**

bool

#### query\_time\_stamp\_setting()

Read time stamp settings (60) from the instrument.

#### wait\_time(wait\_type, N, offset=0)

Configure wait time. (WAT)

#### **Parameters**

- wait\_type (WaitTimeType) Wait time type
- N (float) Coefficient for initial wait time, default: 1
- offset (int, optional) Offset for wait time, defaults to 0

## query\_staircase\_sweep\_settings()

Reads Staircase Sweep Measurement settings (33) from the instrument.

```
sweep_timing(hold, delay, step_delay=0, step_trigger_delay=0, measurement_trigger_delay=0)
```

Sets Hold Time, Delay Time and Step Delay Time for staircase or multi channel sweep measurement. (WT) If not set, all parameters are 0.

#### **Parameters**

- hold (float) Hold time
- **delay** (*float*) Delay time
- **step\_delay** (*float*, *optional*) Step delay time, defaults to 0
- **step\_trigger\_delay** (*float*, *optional*) Trigger delay time, defaults to 0
- measurement\_trigger\_delay (float, optional) Measurement trigger delay time, defaults to 0

## sweep\_auto\_abort(abort, post='START')

Enables/Disables the automatic abort function. Also sets the post measurement condition. (WM)

## **Parameters**

- abort (bool) Enable/Disable automatic abort
- **post** (StaircaseSweepPostOutput, optional) Output after measurement, defaults to 'Start'

#### query\_sampling\_settings()

Reads Sampling Measurement settings (47) from the instrument.

# property sampling\_mode

Set linear or logarithmic sampling mode. (ML)

## **Type**

SamplingMode 5 contract of the sampling of the

## sampling\_timing(hold\_bias, interval, number, hold\_base=0)

Sets Timing Parameters for the Sampling Measurement (MT)

#### **Parameters**

- hold\_bias (float) Bias hold time
- interval (float) Sampling interval
- **number** (*int*) Number of Samples
- hold\_base (float, optional) Base hold time, defaults to 0

```
sampling_auto_abort(abort, post='Bias')
```

Enables/Disables the automatic abort function. Also sets the post measurement condition. (MSC)

#### **Parameters**

- **abort** (*bool*) Enable/Disable automatic abort
- post (SamplingPostOutput, optional) Output after measurement, defaults to 'Bias'

## read\_data(number\_of\_points)

Reads all data from buffer and returns Pandas DataFrame. Specify number of measurement points for correct splitting of the data list.

#### **Parameters**

```
number_of_points (int) - Number of measurement points
```

#### Returns

Measurement Data

## Return type

pd.DataFrame

## read\_channels(nchannels)

Reads data for 1 measurement point from the buffer. Specify number of measurement channels + sweep sources (depending on data output setting).

#### **Parameters**

**nchannels** (int) – Number of channels which return data

#### Returns

Measurement data

## Return type

tuple

# query\_series\_resistor()

Read series resistor status (53) for all SMUs.

## query\_meas\_range\_current\_auto()

Read auto ranging mode status (54) for all SMUs.

# query\_meas\_op\_mode()

Read SMU measurement operation mode (46) for all SMUs.

# query\_meas\_ranges()

Read measruement ranging status (32) for all SMUs.

class pymeasure.instruments.agilent.agilentB1500.SMU(parent, channel, smu\_type, name, \*\*kwargs)

Bases: object

Provides specific methods for the SMUs of the Agilent B1500 mainframe

#### **Parameters**

- parent (AgilentB1500) Instance of the B1500 mainframe class
- **channel** (*int*) Channel number of the SMU
- **smu\_type** (*str*) Type of the SMU
- name (str) Name of the SMU

```
write(string)
     Wraps Instrument.write() method of B1500.
ask(string)
     Wraps ask() method of B1500.
query_learn(query type, command)
     Wraps query_learn() method of B1500.
check_errors()
     Wraps check_errors() method of B1500.
property status
     Query status of the SMU.
enable()
     Enable Source/Measurement Channel (CN)
disable()
    Disable Source/Measurement Channel (CL)
force_gnd()
     Force 0V immediately. Current Settings can be restored with RZ (not implemented). (DZ)
property filter
     Enables/Disables SMU Filter. (FL)
         Type
            bool
property series_resistor
     Enables/Disables 1MOhm series resistor. (SSR)
         Type
            bool
property meas_op_mode
     Set SMU measurement operation mode. (CMM)
         Type
            MeasOpMode
property adc_type
     ADC type of individual measurement channel. (AAD)
         Type
            ADCType
force(source_type, source_range, output, comp=", comp_polarity=", comp_range=")
     Applies DC Current or Voltage from SMU immediately. (DI, DV)
         Parameters
             • source_type (str) – Source type ('Voltage', 'Current')
             • source_range (int or str) – Output range index or name
             • output (float) – Source output value in A or V
             • comp (float, optional) – Compliance value, defaults to previous setting
             • comp_polarity (CompliancePolarity) – Compliance polairty, defaults to auto
```

```
• comp_range (int or str, optional) - Compliance ranging type, defaults to auto
```

```
ramp_source(source_type, source_range, target_output, comp=", comp_polarity=", comp_range=", stepsize=0.001, pause=0.02)
```

Ramps to a target output from the set value with a given step size, each separated by a pause.

#### **Parameters**

- **source\_type** (*str*) Source type ('Voltage' or 'Current')
- target\_output Target output voltage or current
- **irange** (*int*) Output range index
- comp (float, optional) Compliance, defaults to previous setting
- comp\_polarity (CompliancePolarity) Compliance polairty, defaults to auto
- comp\_range (int or str, optional) Compliance ranging type, defaults to auto
- **stepsize** Maximum size of steps
- pause Duration in seconds to wait between steps

# Type

target\_output: float

#### property meas\_range\_current

Current measurement range index. (RI)

Possible settings depend on SMU type, e.g. 0 for Auto Ranging: SMUCurrentRanging

## property meas\_range\_voltage

Voltage measurement range index. (RV)

Possible settings depend on SMU type, e.g. 0 for Auto Ranging: SMUVoltageRanging

## meas\_range\_current\_auto(mode, rate=50)

Specifies the auto range operation. Check Documentation. (RM)

#### **Parameters**

- mode (int) Range changing operation mode
- rate (int, optional) Parameter used to calculate the current value, defaults to 50

staircase\_sweep\_source(source\_type, mode, source\_range, start, stop, steps, comp, Pcomp=")

Specifies Staircase Sweep Source (Current or Voltage) and its parameters. (WV or WI)

#### **Parameters**

- **source\_type** (*str*) Source type ('Voltage', 'Current')
- mode (SweepMode) Sweep mode
- source\_range (int) Source range index
- **start** (*float*) Sweep start value
- stop (float) Sweep stop value
- **steps** (*int*) Number of sweep steps
- comp (float) Compliance value
- Pcomp (float, optional) Power compliance, defaults to not set

```
synchronous_sweep_source(source_type, source_range, start, stop, comp, Pcomp=")
```

Specifies Synchronous Staircase Sweep Source (Current or Voltage) and its parameters. (WSV or WSI)

#### **Parameters**

- **source\_type** (*str*) Source type ('Voltage', 'Current')
- source\_range (int) Source range index
- start (float) Sweep start value
- stop (float) Sweep stop value
- comp (float) Compliance value
- Pcomp (float, optional) Power compliance, defaults to not set

# sampling\_source(source\_type, source\_range, base, bias, comp)

Sets DC Source (Current or Voltage) for sampling measurement. DV/DI commands on the same channel overwrite this setting. (MV or MI)

## **Parameters**

- **source\_type** (*str*) Source type ('Voltage', 'Current')
- source\_range (int) Source range index
- base (float) Base voltage/current
- bias (float) Bias voltage/current
- comp (float) Compliance value

#### **Supporting Classes**

Classes that provide additional functionalities:

- QueryLearn: Process read out of instrument settings
- SMUCurrentRanging, SMUVoltageRanging: Allowed ranges for different SMU types and transformation of range names to indices (base: Ranging)

# class pymeasure.instruments.agilent.agilentB1500.QueryLearn

Bases: object

Methods to issue and process \*LRN? (learn) command and response.

```
static query_learn(ask, query_type)
```

Issues \*LRN? (learn) command to the instrument to read configuration. Returns dictionary of commands and set values.

#### **Parameters**

query\_type (int) - Query type according to the programming guide

#### Returns

Dictionary of command and set values

## **Return type**

dict

# classmethod query\_learn\_header(ask, query\_type, smu\_references, single\_command=False)

Issues \*LRN? (learn) command to the instrument to read configuration. Processes information to human readable values for debugging purposes or file headers.

#### **Parameters**

- ask (Instrument.ask) ask method of the instrument
- query\_type (int or str) Number according to Programming Guide
- **smu\_references** (*dict*) SMU references by channel
- single\_command (str) if only a single command should be returned, defaults to False

#### Returns

Read configuration

## Return type

dict

# static to\_dict(parameters, names, \*args)

Takes parameters returned by *query\_learn()* and ordered list of corresponding parameter names (optional function) and returns dict of parameters including names.

#### **Parameters**

- parameters (dict) Parameters for one command returned by query\_learn()
- names (list) list of names or (name, function) tuples, ordered

#### Returns

Parameter name and (processed) parameter

## Return type

dict

Bases: object

Possible Settings for SMU Current/Voltage Output/Measurement ranges. Transformation of available Voltage/Current Range Names to Index and back.

#### **Parameters**

- **supported\_ranges** (list) Ranges which are supported (list of range indizes)
- ranges (dict) All range names {Name: Indizes}
- **fixed\_ranges** add fixed ranges (negative indizes); defaults to False

```
__call__(input_value)
```

Gives named tuple (name/index) of given Range. Throws error if range is not supported by this SMU.

#### **Parameters**

```
input (str or int) – Range name or index
```

#### Returns

named tuple (name/index) of range

# Return type

namedtuple

class pymeasure.instruments.agilent.agilentB1500.SMUCurrentRanging(smu\_type)

Bases: object

Provides Range Name/Index transformation for current measurement/sourcing. Validity of ranges is checked against the type of the SMU.

Omitting the 'limited auto ranging'/range fixed' specification in the range string for current measurement defaults to 'limited auto ranging'.

Full specification: '1 nA range fixed' or '1 nA limited auto ranging'

'1 nA' defaults to '1 nA limited auto ranging'

class pymeasure.instruments.agilent.agilentB1500.SMUVoltageRanging(smu\_type)

Bases: object

Provides Range Name/Index transformation for voltage measurement/sourcing. Validity of ranges is checked against the type of the SMU.

Omitting the 'limited auto ranging'/range fixed' specification in the range string for voltage measurement defaults to 'limited auto ranging'.

Full specification: '2 V range fixed' or '2 V limited auto ranging'

'2 V' defaults to '2 V limited auto ranging'

## **Enumerations**

Enumerations are used for easy selection of the available parameters (where it is applicable). Methods accept member name or number as input, but name is recommended for readability reasons. The member number is passed to the instrument. Converting an enumeration member into a string gives a title case, whitespace separated string (\_\_str\_\_()) which cannot be used to select an enumeration member again. It's purpose is only logging or documentation.

Bases: IntEnum

Provides additional methods to IntEnum:

- Conversion to string automatically replaces '\_' with ' 'in names and converts to title case
- · get classmethod to get enum reference with name or integer

```
__str__()
```

Gives title case string of enum value

classmethod get(input\_value)

Gives Enum member by specifying name or value.

```
Parameters
```

```
input_value (str or int) - Enum name or value
```

Returns

Enum member

Bases: CustomIntEnum

ADC Type

HSADC = 0

High-speed ADC

```
HRADC = 1
         High-resolution ADC
     HSADC PULSED = 2
         High-resolution ADC for pulsed measurements
class pymeasure.instruments.agilent.agilentB1500.ADCMode(value, names=<not given>, *values,
                                                              module=None, qualname=None,
                                                              type=None, start=1, boundary=None)
     Bases: CustomIntEnum
     ADC Mode
     AUTO = 0
     MANUAL = 1
     PLC = 2
     TIME = 3
class pymeasure.instruments.agilent.agilentB1500.AutoManual(value, names=<not given>, *values,
                                                                 module=None, qualname=None,
                                                                 type=None, start=1,
                                                                 boundary=None)
     Bases: CustomIntEnum
     Auto/Manual selection
     AUTO = 0
     MANUAL = 1
class pymeasure.instruments.agilent.agilentB1500.MeasMode(value, names=<not given>, *values,
                                                               module=None, qualname=None,
                                                               type=None, start=1, boundary=None)
     Bases: CustomIntEnum
     Measurement Mode
     SPOT = 1
     STAIRCASE\_SWEEP = 2
     SAMPLING = 10
class pymeasure.instruments.agilent.agilentB1500.MeasOpMode(value, names=<not given>, *values,
                                                                 module=None, qualname=None,
                                                                 type=None, start=1,
                                                                 boundary=None)
     Bases: CustomIntEnum
     Measurement Operation Mode
     COMPLIANCE_SIDE = 0
     CURRENT = 1
     VOLTAGE = 2
```

```
FORCE\_SIDE = 3
     COMPLIANCE_AND_FORCE_SIDE = 4
class pymeasure.instruments.agilent.agilentB1500.SweepMode(value, names=<not given>, *values,
                                                                 module=None, qualname=None,
                                                                 type=None, start=1, boundary=None)
     Bases: CustomIntEnum
     Sweep Mode
     LINEAR_SINGLE = 1
     LOG_SINGLE = 2
     LINEAR_DOUBLE = 3
     LOG_DOUBLE = 4
class pymeasure.instruments.agilent.agilentB1500.SamplingMode(value, names=<not given>,
                                                                     *values, module=None,
                                                                     qualname=None, type=None,
                                                                     start=1, boundary=None)
     Bases: CustomIntEnum
     Sampling Mode
     LINEAR = 1
     LOG_10 = 2
         Logarithmic 10 data points/decade
     LOG_25 = 3
         Logarithmic 25 data points/decade
     LOG_50 = 4
         Logarithmic 50 data points/decade
     LOG 100 = 5
         Logarithmic 100 data points/decade
     LOG_250 = 6
         Logarithmic 250 data points/decade
     LOG_{5000} = 7
         Logarithmic 5000 data points/decade
class pymeasure.instruments.agilent.agilentB1500.SamplingPostOutput(value, names=<not given>,
                                                                            *values, module=None,
                                                                            qualname=None,
                                                                            type=None, start=1,
                                                                            boundary=None)
     Bases: CustomIntEnum
     Output after sampling
     BASE = 1
     BIAS = 2
```

```
class pymeasure.instruments.agilent.agilentB1500.StaircaseSweepPostOutput(value, names=<not</pre>
                                                                                 given>, *values,
                                                                                 module=None,
                                                                                 qualname=None,
                                                                                 type=None,
                                                                                 start=1,
                                                                                 boundary=None)
     Bases: CustomIntEnum
     Output after staircase sweep
     START = 1
     STOP = 2
class pymeasure.instruments.agilent.agilentB1500.CompliancePolarity(value, names=<not given>,
                                                                           *values, module=None,
                                                                          qualname=None,
                                                                          type=None, start=1,
                                                                          boundary=None)
     Bases: CustomIntEnum
     Compliance polarity
     AUTO = 0
     MANUAL = 1
class pymeasure.instruments.agilent.agilentB1500.WaitTimeType(value, names=<not given>,
                                                                    *values, module=None,
                                                                    qualname=None, type=None,
                                                                    start=1, boundary=None)
     Bases: CustomIntEnum
     Wait time type
     SMU_SOURCE = 1
     SMU\_MEASUREMENT = 2
     CMU\_MEASUREMENT = 3
7.9.14 Agilent 4284A LCR Meter
class pymeasure.instruments.agilent.Agilent4284A(adapter, name='Agilent 4284A LCR meter',
                                                      **kwargs)
     Bases: SCPIMixin, Instrument
```

Represents the Agilent 4284A precision LCR meter.

```
agilent = Agilent4284A("GPIB::1::INSTR")

agilent.reset()  # Return instrument settings to default values

agilent.frequency = 10e3  # Set frequency to 10 kHz

agilent.voltage = 0.02  # Set AC voltage to 20 mV

(continues on next page)
```

(continued from previous page)

```
agilent.mode = 'ZTR'
                                        # Set impedance mode to measure
                                          impedance magnitude [Ohm] and phase
                                          [rad]
agilent.sweep_measurement(
                                        # Perform frequency sweep measurement
    'frequency', [1e4, 1e3, 100]
                                        # at 10 kHz, 1 kHz, and 100 Hz
                                        # Enable upper current, voltage, and
agilent.enable_high_power()
                                          bias limits, if properly configured.
agilent.correction.open_enabled = True
                                           # Enable the OPEN correction
agilent.correction.spot1.enabled = True
                                           # Enable the correction for SPOT1
agilent.correction.spot1.frequency = 10e3 # Set SPOT1 frequency to 10 kHz
agilent.correction.spot1.measure_open()
                                          # Measure the OPEN structure for SPOT1
```

#### correction

#### Channel

Agilent4284ACorrection

#### property ac\_current

Control AC current level in Amps. Valid range is 50 uA to 20 mA for default, 50 uA to 200 mA in high-power mode.(dynamic)

## property ac\_voltage

Control AC voltage level in Volts. Range is 5 mV to 2 V for default, 5 mV to 20 V in high-power mode.(dynamic)

# property auto\_range\_enabled

Control whether the impedance auto range is enabled.

## property bias\_current

Control the DC bias current in Amps. Requires Option 001 (power amplifier / DC bias) to be installed. Maximum is 100 mA.(dynamic)

# property bias\_enabled

Control whether DC bias is enabled.

## property bias\_voltage

Control the DC bias voltage in Volts. Maximum is 2 V by default, 40 V in high-power mode.(dynamic)

#### property frequency

Control AC frequency in Hertz, from 20 Hz to 1 MHz.(dynamic)

# property high\_power\_enabled

Control whether high power mode is enabled. Enabling requires option 001 (power amplifier / DC bias) to be installed.

## property impedance\_mode

Control impedance measurement function.

- CPD: Parallel capacitance [F] and dissipation factor [number]
- CPQ: Parallel capacitance [F] and quality factor [number]
- CPG: Parallel capacitance [F] and parallel conductance [S]
- CPRP: Parallel capacitance [F] and parallel resistance [Ohm]

- CSD: Series capacitance [F] and dissipation factor [number]
- CSQ: Series capacitance [F] and quality factor [number]
- CSRS: Series capacitance [F] and series resistance [Ohm]
- LPQ: Parallel inductance [H] and quality factor [number]
- LPD: Parallel inductance [H] and dissipation factor [number]
- LPG: Parallel inductance [H] and parallel conductance [S]
- LPRP: Parallel inductance [H] and parallel resistance [Ohm]
- LSD: Series inductance [H] and dissipation factor [number]
- LSQ: Seriesinductance [H] and quality factor [number]
- LSRS: Series inductance [H] and series resistance [Ohm]
- RX: Resistance [Ohm] and reactance [Ohm]
- ZTD: Impedance, magnitude [Ohm] and phase [deg]
- ZTR: Impedance, magnitude [Ohm] and phase [rad]
- GB: Conductance [S] and susceptance [S]
- YTD: Admittance, magnitude [Ohm] and phase [deg]
- YTR: Admittance magnitude [Ohm] and phase [rad]

#### property impedance\_range

Control the impedance measurement range. The 4284A will select an appropriate measurement range for the setting value.

# sweep\_measurement(sweep\_mode, sweep\_values)

Run list sweep measurement using sequential trigger.

#### **Parameters**

- **sweep\_mode** (*str*) parameter to sweep across. Must be one of *frequency*, *voltage*, *current*, *bias\_voltage*, or *bias\_current*.
- **sweep\_values** list of parameter values to sweep across.

#### Returns

values as configured with *impedance\_mode* and list of sweep parameters in format ([val A], [val B], [sweep\_values])

#### trigger()

Execute a bus trigger, regardless of trigger state. Can be used when *trigger\_source* is set to *BUS*. Returns result of triggered measurement.

# property trigger\_continuous\_enabled

Control whether trigger state automatically returns to WAIT FOR TRIGGER after measurement.

#### property trigger\_delay

Control trigger delay in seconds. Valid range is 0 to 60, with 1 ms resolution.

#### trigger\_immediate()

Execute a bus trigger, regardless of trigger state. Can be used when *trigger\_source* is set to *BUS*. Measurement result must be retrieved with *FETCH*? command.

```
trigger_initiate()
```

Change the trigger state from IDLE to WAIT FOR TRIGGER for one trigger sequence.

## property trigger\_source

Control trigger mode. Valid options are INT, EXT, BUS, or HOLD.

#### class pymeasure.instruments.agilent.agilent4284A.Agilent4284ACorrection(parent, id)

Bases: Channel

A class representing the correction functions.

## property cable\_length

Control the correction setting for cable length in meters, (int strictly 0, 1, 2 or 4).

## property load\_enabled

Control the LOAD correction (bool).

# property load\_function

Control the spot LOAD standard.

#### Type

str, strictly in CPD, CPQ, CPG, CPRP, CSD, CSQ, CSRS, LPQ, LPD, LPG, LPRP, LSD, LSQ, LSRS, RX, ZTD, ZTR, GB, YTD, YTR

See Agilent4284A.impedance\_mode for detailed explanation.

#### measure\_load()

Measure the LOAD correction standard.

#### measure\_open()

Measure the OPEN correction standard.

## measure\_short()

Measure the SHORT correction standard.

## property open\_enabled

Control the OPEN correction (bool).

# property short\_enabled

Control the SHORT correction (bool).

## class pymeasure.instruments.agilent.agilent4284A.Agilent4284ASpot(parent, id)

Bases: Channel

A class representing the spot correction functions.

# property enabled

Control this spot correction (bool).

#### property frequency

Control the frequency of this spot in Hz.

## property load\_function

Control the LOAD standard of this spot.

## Type

str, strictly in CPD, CPQ, CPG, CPRP, CSD, CSQ, CSRS, LPQ, LPD, LPG, LPRP, LSD, LSQ, LSRS, RX, ZTD, ZTR, GB, YTD, YTR

See Agilent4284A.impedance\_mode for detailed explanation.

## measure\_load()

Measure the LOAD correction standard of this spot.

## measure\_open()

Measure the OPEN correction standard of this spot.

#### measure\_short()

Measure the SHORT correction standard of this spot.

# 7.9.15 Agilent B2980 Series, Femto/Picoammeter Electrometer/High Resistance Meter

Bases: SCPIMixin, Instrument

A class representing the Agilent/Keysight B2981A/B.

The B2981 is a Femto/Picoammeter.

#### abort()

Abort the all actions.

## abort\_acquisition()

Abort action 'ACQuire'.

#### arm()

Send an immediate arm trigger for all actions.

When the status of all actions is initiated, the arm trigger causes a layer change from arm to trigger.

## arm\_acquisition()

Send an immediate arm trigger for action 'ACQuire'.

When the status of action 'ACQuire' is initiated, the arm trigger causes a layer change from arm to trigger.

## property arm\_acquisition\_bypass\_once\_enabled

Control the bypass for the event detector in the arm layer for action 'ACQuire' (bool).

# property arm\_acquisition\_count

Control the arm trigger count for action 'ACQuire'.

#### **Type**

- int, strictly from 1 to 100000 or
- str, strictly in MIN, MAX, DEF, INF

INF is equivalent to 2147483647

## property arm\_acquisition\_delay

Control the arm trigger delay for action 'ACQuire' in seconds.

## **Type**

- float, strictly from 0 to 1E5 or
- str, strictly in MIN, MAX, DEF

#### property arm\_acquisition\_output\_enabled

Control the arm trigger output for action 'ACQuire' (bool).

It is for the status change between the idle state and the arm layer.

#### property arm\_acquisition\_output\_signal

Control the arm trigger output for action 'ACQuire'.

## Type

str, strictly in INT1, INT2, LAN, TOUT, EXT1 to EXT7

- INT1 or INT2 selects the internal bus 1 or 2.
- LAN selects a LAN port.
- EXT1 to EXT7 selects the GPIO pin n, which is an output port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TOUT selects the BNC Trigger Out.

It is for the status change between the idle state and the arm layer.

#### property arm\_acquisition\_source

Control the arm trigger source for action 'ACQuire'.

#### Type

str, strictly in AINT, BUS, TIM, INT1, INT2, LAN, TIN, EXT1 to EXT7

- AINT automatically selects the arm source most suitable for the present operating mode by using internal algorithms.
- BUS selects the remote interface trigger command such as the group execute trigger (GET) and the TRG command.
- TIM selects a signal internally generated every interval set by the arm\_timer command.
- INT1 and INT2 selects a signal from the internal bus 1 or 2.
- LAN selects the LXI trigger specified by the arm\_source\_lan\_id command.
- EXT1 to EXT7 selects a signal from the GPIO pin n, which is an input port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TIN selects the BNC Trigger In.

## property arm\_acquisition\_source\_lan\_id

Control the source for LAN arm triggers for action 'ACQuire'.

#### **Type**

str, strictly from LANO to LAN7

## property arm\_acquisition\_timer

Control the timer interval of the arm trigger source for action 'ACQuire' in seconds.

# Type

- float, strictly from 1E-5 to 1E5
- str, strictly in MIN, MAX, DEF

# property arm\_all\_bypass\_once\_enabled

Set the bypass for the event detector in the arm layer for action 'ALL' (bool).

#### property arm\_all\_count

Set the arm trigger counter for action 'ALL'.

# **Type**

- int, strictly from 1 to 100000 or
- str, strictly in MIN, MAX, DEF, INF

INF is equivalent to 2147483647.

## property arm\_all\_delay

Set the arm trigger delay for action 'ALL' in seconds.

## **Type**

- float, strictly from 0 to 1E5 or
- str, strictly in MIN, MAX, DEF

# property arm\_all\_output\_enabled

Set the arm trigger output for action 'ALL' (bool).

It is for the status change between the idle state and the arm layer.

## property arm\_all\_output\_signal

Set the arm trigger output for action 'ALL'.

#### Type

str, strictly in INT1, INT2, LAN, TOUT, EXT1 to EXT7

- INT1 and INT2 selects the internal bus 1 or 2.
- · LAN selects a LAN port.
- EXT1 to EXT7 selects the GPIO pin n, which is an output port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TOUT selects the BNC Trigger Out.

It is for the status change between the idle state and the arm layer.

## property arm\_all\_source

Set the arm trigger source for action 'ALL'.

#### **Type**

str, strictly in AINT, BUS, TIM, INT1, INT2, LAN, TIN, EXT1 to EXT7

- AINT automatically selects the arm source most suitable for the present operating mode by using internal algorithms.
- BUS selects the remote interface trigger command such as the group execute trigger (GET) and the TRG command.
- TIM selects a signal internally generated every interval set by attr:arm\_all\_timer.
- INT1 and INT2 selects a signal from the internal bus 1 or 2.
- LAN selects the LXI trigger specified by arm\_all\_source\_lan\_id.
- EXT1 to EXT7 selects a signal from the GPIO pin n, which is an input port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TIN selects the BNC Trigger In.

## property arm\_all\_source\_lan\_id

Set the source for LAN arm triggers for action 'ALL'.

# Type

str, strictly from LANO to LAN7

## property arm\_all\_timer

Set the timer interval of the arm trigger source for action 'ALL' in seconds.

## **Type**

- float, strictly from 1E-5 to 1E5 or
- str, strictly in MIN, MAX, DEF

## property current

Measure current with a spot measurement in Amps.

## Returns

float

# property current\_range

Control the range for the current measurement in Amps.

# Type

- float, strictly from 2E-12 to 20E-3 or
- str, strictly in MIN, MAX, DEF, UP, DOWN

## property data\_buffer\_size

Get the data buffer size (int).

# init()

Initiate a trigger for all actions.

## init\_acquisition()

Initiate a trigger for action 'ACQuire'.

## property input\_enabled

Control the instrument input relay (bool).

## property interlock\_enabled

Get the interlock status (bool).

#### property measure

Measure the defined parameter(s) with a spot measurement.

#### Returns

float or list of float

# property trigger\_acquisition\_bypass\_once\_enabled

Control the bypass for the event detector in the trigger layer for action 'ACQuire' (bool).

# property trigger\_acquisition\_count

Control the trigger count for action 'ACQuire'.

# **Type**

- int, strictly from 1 to 100000 or
- str, strictly in MIN, MAX, DEF, INF

INF is equivalent to 2147483647.

## property trigger\_acquisition\_delay

Control the trigger delay for action 'ACQuire' in seconds.

#### Type

- float, strictly from 0 to 1E5 or
- str, strictly in MIN, MAX, DEF

## property trigger\_acquisition\_is\_idle

Get the idle status of action 'ACQuire' (bool).

The command waits until the status is changed to idle.

## property trigger\_acquisition\_output\_enabled

Control the trigger output for action 'ACQuire' (bool).

It is for the status change between the idle state and the trigger layer.

## property trigger\_acquisition\_output\_signal

Control the trigger signal output for action 'ACQuire'.

## Type

str, strictly in INT1, INT2, LAN, TOUT, EXT1 to EXT7

- INT1 and INT2 selects the internal bus 1 or 2.
- LAN selects a LAN port.
- EXT1 to EXT7 selects the GPIO pin n, which is an output port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TOUT: selects the BNC Trigger Out.

It is for the status change between the idle state and the trigger layer.

## property trigger\_acquisition\_source

Control the trigger source for action 'ACQuire'.

# **Type**

str, strictly in AINT, BUS, TIM, INT1, INT2, LAN, TIN, EXT1 to EXT7

- AINT automatically selects the trigger source most suitable for the present operating mode by using internal algorithms.
- BUS selects the remote interface trigger command such as the group execute trigger (GET) and the TRG command.
- TIM selects a signal internally generated every interval set by the trigger\_acquisition\_timer.
- INT1 and INT2 selects a signal from the internal bus 1 or 2.
- LAN selects the LXI trigger specified by trigger\_acquisition\_source\_lan\_id.
- EXT1 to EXT7 selects a signal from the GPIO pin n, which is an input port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TIN selects the BNC Trigger In.

#### property trigger\_acquisition\_source\_lan\_id

Control the source for LAN triggers for action 'ACQuire'.

## **Type**

str, strictly from LANO to LAN7

#### property trigger\_acquisition\_timer

Control the timer interval of the trigger source for action 'ACQuire' in seconds.

## **Type**

- float, strictly from 1E-5 to 1E5 or
- str, strictly in MIN, MAX, DEF

## property trigger\_all\_bypass\_once\_enabled

Set the bypass for the event detector in the trigger layer for action 'ALL' (bool).

# property trigger\_all\_count

Set the trigger count for action 'ALL'.

## **Type**

- int, strictly from 1 to 100000 or
- str, strictly in MIN, MAX, DEF, INF

INF is equivalent to 2147483647.

# property trigger\_all\_delay

Set the trigger delay for action 'ALL' in seconds.

## **Type**

- float, strictly from 0 to 1E5 or
- str, strictly in MIN, MAX, DEF

# property trigger\_all\_is\_idle

Get the trigger idle status for action 'ALL' (bool).

The command waits until the status is changed to idle.

# property trigger\_all\_output\_enabled

Set the trigger output for action 'ALL' (bool).

It is for the status change between the idle state and the trigger layer.

## property trigger\_all\_output\_signal

Set the trigger signal output for action 'ALL'.

#### Type

str, strictly in INT1, INT2, LAN, TOUT, EXT1 to EXT7

- INT1 and INT2 selects the internal bus 1 or 2.
- LAN selects a LAN port.
- EXT1 to EXT7 selects the GPIO pin n, which is an output port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TOUT selects the BNC Trigger Out.

It is for the status change between the idle state and the trigger layer.

#### property trigger\_all\_source

Set the trigger source for action 'ALL'.

## **Type**

str, strictly in AINT, BUS, TIM, INT1, INT2, LAN, TIN, EXT1 to EXT7

- AINT automatically selects the trigger source most suitable for the present operating mode by using internal algorithms.
- BUS selects the remote interface trigger command such as the group execute trigger (GET) and the TRG command.
- TIM selects a signal internally generated every interval set by trigger\_all\_timer.
- INT1 and INT2 selects a signal from the internal bus 1 or 2.
- LAN selects the LXI trigger specified by trigger\_all\_source\_lan\_id.
- EXT1 to EXT7 selects a signal from the GPIO pin n, which is an input port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TIN selects the BNC Trigger In.

# property trigger\_all\_source\_lan\_id

Set the source for LAN triggers for action 'ALL'.

## **Type**

str, strictly from LANO to LAN7

## property trigger\_all\_timer

Set the timer interval of the trigger source for action 'ALL' in seconds.

## **Type**

- float, strictly from 1E-5 to 1E5 or
- str, strictly in MIN, MAX, DEF

#### property zero\_corrected

Control the zero correction function (bool).

Bases: AgilentB2981, BatteryMixin

A class representing the Agilent/Keysight B2983A/B.

The B2983 is a Femto/Picoammeterwith with battery operation.

Bases: AgilentB2981

A class representing the Agilent/Keysight B2985A/B.

The B2985 is a Femto/Picoammeter and Electrometer/High Resistance Meter.

# abort\_transient()

Abort action 'TRANSient'.

#### arm\_transient()

Send an immediate arm trigger for action 'TRANSient'.

When the status of the specified action is initiated, the arm trigger causes a layer change from arm to trigger.

#### property arm\_transient\_bypass\_once\_enabled

Control the bypass for the event detector in the arm layer for action 'TRANSient' (bool).

#### property arm\_transient\_count

Control the arm trigger count for action 'TRANSient'.

# Type

- int, strictly from 1 to 100000 or
- str, strictly in MIN, MAX, DEF, INF

INF is equivalent to 2147483647.

## property arm\_transient\_delay

Control the arm trigger delay for action 'TRANSient' in seconds.

#### Type

- float, strictly from 0 to 1E5 or
- str, strictly in MIN, MAX, DEF

## property arm\_transient\_output\_enabled

Control the arm trigger output for action 'TRANSient' (bool).

It is for the status change between the idle state and the arm layer.

## property arm\_transient\_output\_signal

Control the arm trigger output for action 'TRANSient'.

#### **Type**

str, strictly in INT1, INT2, LAN, TOUT, EXT1 to EXT7

- INT1 and INT2 selects the internal bus 1 or 2.
- LAN selects a LAN port.
- EXT1 to EXT7 selects the GPIO pin n, which is an output port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TOUT selects the BNC Trigger Out.

It is for the status change between the idle state and the arm layer.

## property arm\_transient\_source

Control the arm trigger source for action 'TRANSient'.

## **Type**

str, strictly in AINT, BUS, TIM, INT1, INT2, LAN, TIN, EXT1 to EXT7  $\,$ 

- AINT automatically selects the arm source most suitable for the present operating mode by using internal algorithms.
- BUS selects the remote interface trigger command such as the group execute trigger (GET) and the TRG command.
- TIM selects a signal internally generated every interval set by arm\_transient\_timer.

- INT1 and INT2 selects a signal from the internal bus 1 or 2.
- LAN selects the LXI trigger specified by arm\_transient\_source\_lan\_id.
- EXT1 to EXT7 selects a signal from the GPIO pin n, which is an input port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TIN selects the BNC Trigger In.

## property arm\_transient\_source\_lan\_id

Control the source for LAN arm triggers for action 'TRANSient'.

#### **Type**

str, strictly from LANO to LAN7

# property arm\_transient\_timer

Control the timer interval of the arm trigger source for action 'TRANSient' in seconds.

# Type

- float, strictly from 1E-5 to 1E5
- str, strictly in MIN, MAX, DEF

# property charge

Measure charge with a spot measurement in Coulombs.

#### Returns

float

# property charge\_range

Control the range for the charge measurement in Coulombs.

# Type

- float, strictly from 2E-9 to 2E-6 or
- str, strictly in MIN, MAX, DEF, UP, DOWN

## property function

Control the measurement function.

# **Type**

str, strictly in CURR, CHAR, VOLT, RES

#### property humidity

Measure the relative humidity in percent.

#### Returns

float

# init\_transient()

Initiate a trigger for action 'TRANSient'.

## property resistance

Measure resistance with a spot measurement in Ohms.

# Returns

float

# property resistance\_range

Control the range for the resistance measurement.

# Type

- float, strictly from 1E6 to 1E15 or
- str, strictly in MIN, MAX, DEF, UP, DOWN

# property source\_enabled

Control the voltage source output relay (bool).

# property source\_low\_state

Control the voltage source low terminal state.

## **Type**

str, strictly FLO or COMM

- FLO: low terminal is floating.
- COMM: low terminal is connected to Common.

# property source\_off\_state

Control the voltage source off condition.

# Type

str, strictly in ZERO, HIZ, NORM

HIGH Z	<ul> <li>Output relay: off (open)</li> <li>The voltage source setting is not changed.</li> <li>This status is available only when the 20 V range is used.</li> </ul>
NORMAL	<ul><li>Output voltage: 0 V</li><li>Output relay: off (open)</li></ul>
ZERO	• Output voltage: 0 V in the present voltage range

# property source\_voltage

Control the source voltage in Volts.

#### **Type**

float, strictly from -1050 to 1050

For voltages > 20V the interlock must be closed.

Range	Resolution	Output voltage	Maximum current
20 V	$700  \mu V$	-21 V V 21 V	+/-20 mA
1000 V	35 mV	-1 V V 1000 V	+/-1.0 mA
-1000 V	35 mV	-1000 V V 1 V	+/-1.0 mA

## property source\_voltage\_range

Control the source voltage range.

# Type

- float, strictly from -1000 to 1000 or
- str, strictly in MIN, MAX, DEF

## property temperature

Measure the temperature.

#### **Returns**

float

The unit of temperature is controlled by temperature\_unit:

## property temperature\_sensor

Control the tempertature sensor.

## **Type**

str, strictly in TC, HSEN

- TC selects the thermocouple.
- HSEN selects the temperature sensor within the humidity sensor.

# property temperature\_unit

Control the tempertature unit.

## **Type**

str, strictly in C, CEL, F, FAR, K

- C, CEL selects degree Celsius.
- F, FAR selects degree Fahrenheit.
- · K selects Kelvin.

# property trigger\_transient\_bypass\_once\_enabled

Control the bypass for the event detector in trigger layer for action 'TRANSient' (bool).

# property trigger\_transient\_count

Control the trigger count for action 'TRANSient'.

#### **Type**

- int, strictly from 1 to 100000 or
- str, strictly in MIN, MAX, DEF, INF

INF is equivalent to 2147483647.

## property trigger\_transient\_delay

Control the trigger delay for action 'TRANSient' in seconds.

# **Type**

- float, strictly from 0 to 1E5 or
- str, strictly in MIN, MAX, DEF

7.9. Agilent 211

# property trigger\_transient\_is\_idle

Get idle the status of action 'TRANSient' (bool).

The command waits until the status is changed to idle.

# property trigger\_transient\_output\_enabled

Control the trigger output for action 'TRANSient' (bool).

It is for the status change between the idle state and the trigger layer.

# property trigger\_transient\_output\_signal

Control the trigger signal output for action 'TRANSient'.

### **Type**

str, strictly in INT1, INT2, LAN, TOUT, EXT1 to EXT7

- INT1, INT2 selects the internal bus 1 or 2.
- LAN selects a LAN port.
- EXT1 to EXT7 selects the GPIO pin n, which is an output port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TOUT selects the BNC Trigger Out.

It is for the status change between the idle state and the trigger layer.

# property trigger\_transient\_source

Control the trigger source for action 'TRANSient'.

# **Type**

str, strictly in AINT, BUS, TIM, INT1, INT2, LAN, TIN, EXT1 to EXT7

- AINT automatically selects the trigger source most suitable for the present operating mode by using internal algorithms.
- BUS selects the remote interface trigger command such as the group execute trigger (GET) and the TRG command.
- TIM selects a signal internally generated every interval set by trigger\_transient\_timer.
- INT1 and INT2 selects a signal from the internal bus 1 or 2.
- LAN selects the LXI trigger specified by trigger\_transient\_source\_lan\_id.
- EXT1 to EXT7 selects a signal from the GPIO pin n, which is an input port of the Digital I/O D-sub connector on the rear panel. n = 1 to 7.
- TIN selects the BNC Trigger In.

# property trigger\_transient\_source\_lan\_id

Control the source for LAN triggers for action 'TRANSient'.

#### **Type**

str, strictly from LANO to LAN7

# property trigger\_transient\_timer

Control the timer interval of the trigger source for action 'TRANSient'.

## **Type**

• float, strictly from 1E-5 to 1E5 or

• str, strictly in MIN, MAX, DEF

# property voltage

Measure voltage with a spot (one-shot) measurement in Volts.

#### Returns

float

# property voltage\_range

Control the range for the voltage measurement in Volts.

# **Type**

- float, strictly from 2 to 20 or
- str, strictly in MIN, MAX, DEF, UP, DOWN

Bases: AgilentB2985, BatteryMixin

A class representing the Agilent/Keysight B2987A/B.

The B2987 is a Femto/Picoammeter and Electrometer/High Resistance Meter with battery operation.

# class pymeasure.instruments.agilent.agilentB298x.BatteryMixin

Bases: object

A class representing the B2983/7 battery functions.

# property battery\_cycles

Get the battery cycle count (int).

# property battery\_level

Get the percentage of the remaining battery capacity (int).

# property battery\_selftest\_passed

Get the battery self-test result (bool).

# 7.9.16 Agilent E5270B SMU mainframe

class pymeasure.instruments.agilent.AgilentE5270B(adapter, name='Agilent E5270B', \*\*kwargs)

Bases: SCPIMixin, Instrument

A class representing the Agilent E5270B 8 slot SMU mainframe.

It supports the following plug-in modules:

Name	Туре	Slots	Range	Resolution
E5280B	HPSMU	2	$\pm 200 \text{ V}, \pm 1 \text{ A}$	2 μV, 10 fA
E5281B	MPSMU	1	$\pm 100$ V, $\pm 100$ mA	0.5 μV, 10 fA
E5287A	HRSMU	1	$\pm 100 \text{ V}, \pm 100 \text{ mA}$	0.5 μV, 1 fA
E5288A	ASU	_	$\pm 100$ V, $\pm 100$ mA	$0.5 \mu V$ , $0.1 fA$

# check\_errors()

Read all errors from the instrument.

7.9. Agilent 213

#### Returns

List of error entries.

# get\_error\_message(error\_code)

Return the error message for to the specified error code (str).

### property options

Get the installed SMUs (list of str).

Each list entry contains the module name and its revision, e.g. E5281B, 0 An empty slot returns 0,0

class pymeasure.instruments.agilent.agilentE5270B.SMUChannel(parent, id)

Bases: Channel

A class representing the Agilent E5270B SMU channel.

## property current

Measure the current in Amps (float).

### property current\_setpoint

Set range, output current and voltage compliance (int, float, float).

```
inst = AgilentE5270B("GPIB0::17::INSTR")
inst.smu1.current_setpoint = (0, 1.32e-3, 0.5) # (range, value, compliance)
# Set SMU1 to output 1.32 mA in autorange with 0.5 V voltage compliance
```

### Ranges:

- 0: Auto ranging
- 8: 1 pA limited auto ranging for E5287A+E5288A
- 9: 10 pA limited auto ranging for E5287A
- 10: 100 pA limited auto ranging for E5287A
- 11: 1 nA limited auto ranging
- 12: 10 nA limited auto ranging
- 13: 100 nA limited auto ranging
- 14: 1 A limited auto ranging
- 15: 10 A limited auto ranging
- 16: 100 A limited auto ranging
- 17: 1 mA limited auto ranging
- 18: 10 mA limited auto ranging
- 19: 100 mA limited auto ranging
- 20: 1 A limited auto ranging for E5280B

### property enabled

Set the channel output state (bool).

## property voltage

Measure the voltage in Volts (float).

### property voltage\_setpoint

Set range, output voltage and current compliance (int, float, float)

```
inst = AgilentE5270B("GPIB0::17::INSTR")
inst.smu1.voltage_setpoint = (12, 14.42, 0.05) # (range, value, compliance)
# Set SMU1 to output 14.42 V in 20 V range with 50 mA current compliance
```

### Ranges:

- 0: Auto ranging
- 5: 0.5 V limited auto ranging for E5281B/E5287A
- 50: 5 V limited auto ranging for E5281B/E5287A
- 11 or 20: 2 V limited auto ranging
- 12 or 200: 20 V limited auto ranging
- 13 or 400: 40 V limited auto ranging
- 14 or 1000: 100 V limited auto ranging
- 15 or 2000: 200 V limited auto ranging for E5280B

# 7.10 Aim-TTI

This section contains specific documentation on the Aim-TTI instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.10.1 Aim-TTI PL Series Power Supplies

```
class pymeasure.instruments.aimtti.aimttiPL.PLBase(adapter, name='AimTTI PL', **kwargs)
```

Bases: SCPIUnknownMixin, Instrument

Control AimTTI PL series power supplies. Model number ending with -P or P(G) support this remote interface.

#### **Documentation:**

https://resources.aimtti.com/manuals/New\_PL+PL-P\_Series\_Instruction\_Manual-Iss18.pdf

#### PL-series devices:

https://www.aimtti.com/product-category/dc-power-supplies/aim-plseries

The default value for the timeout argument is set to 5000ms.

```
psu = PL303QMDP("ASRL7::INSTR")
psu.reset()
psu.ch_2.voltage = 1.2
psu.ch_2.output_enabled = True
...
psu.ch_2.output_enabled = False
psu.local()
```

# property all\_outputs\_enabled

Control whether all sources are enabled simultaneously, takes values True or False.

7.10. Aim-TTI 215

```
local()
```

Go to local. Make sure all output are disabled first.

Bases: Channel

A channel of AimTTI PL series power supplies.

Channels of the power supply. The channels are number from right-to-left, starting at 1.

## property current

Measure the output readback current for this output channel in Amps.

# property current\_limit

Control the current limit in Amps.(dynamic)

# property current\_range

Control the current range of the channel. Low (500/800mA) range, or High range. Output must be switched off before changing range.

# property output\_enabled

Control whether the source is enabled, takes values True or False.

### property voltage

Measure the output readback voltage for this output channel in Volts.

# property voltage\_setpoint

Control the output voltage of this channel. With verify: the operation is completed when the parameter being adjusted reaches the required value to within  $\pm 5\%$  or  $\pm 10$  counts.(dynamic)

**class** pymeasure.instruments.aimtti.laimttiPL.**PL068P**(adapter, name='AimTTI PL068-P', \*\*kwargs)

Bases: PLBase

ch 1

### Channel

**PLChannel** 

class pymeasure.instruments.aimtti.aimttiPL.PL155P(adapter, name='AimTTI PL155-P', \*\*kwargs)

Bases: PLBase

 $ch_1$ 

#### Channel

**PLChannel** 

class pymeasure.instruments.aimtti.aimttiPL.PL303P(adapter, name='AimTTI PL303-P', \*\*kwargs)

Bases: PLBase

ch\_1

#### Channel

**PLChannel** 

class pymeasure.instruments.aimtti.aimttiPL.PL601P(adapter, name='AimTTI PL601-P', \*\*kwargs)

Bases: PLBase

```
ch_1
             Channel
                PLChannel
class pymeasure.instruments.aimtti.aimttiPL.PL303QMDP(adapter, name='AimTTI PL303QMD-P',
                                                          **kwargs)
     Bases: PLBase
     ch 1
             Channel
                PI.Channel
     ch 2
             Channel
                PLChannel
class pymeasure.instruments.aimtti.aimttiPL.PL303QMTP(adapter, name='AimTTI PL303QMT-P',
                                                          **kwargs)
     Bases: PLBase
     ch_1
             Channel
                PLChannel
     ch_2
             Channel
                PLChannel
     ch_3
             Channel
                PLChannel
```

# 7.10.2 Aim-TTi Electronic Loads

```
class pymeasure.instruments.aimtti.ld400p.LD400P(adapter, name='LD400', **kwargs)
    Bases: SCPIMixin, Instrument
    Interface for the LD400P electronic DC load from Aim-TTi.
    Note: the LD400 edition doesn't have digital interfacing, only the LD400P does.
    The interface is described by the LD400[P] manual.
    The class SCPIMixin is inherited, however the commands options, next_error` and check_errors do not work for this device.
    check_errors()
        Not available.
    property current
        Get the measured load current (float).
    property input_enabled
```

7.10. Aim-TTI 217

Control the input state of the digital load, can be True (on) or False (off).

#### property level\_a

Control the A level of the current control mode (float).

#### property level\_b

Control the B level of the current control mode (float).

#### property level\_select

Control the selected level of the digital load. Can be one of: ("A", "B", "T", "V", "E"), corresponding to (respectively) Level A, Level B, Transient, Ext Voltage and Ext TTL (string).

### property mode

Control the mode of the digital load. Can be one of: ("C", "P", "R", "G", "V"), corresponding to (respectively) constant current, power, resistance, conductance, or voltage (string).

### property next\_error

Get next error - Not available.

#### property options

Get options - Not available.

### classmethod remove\_unit\_suffix(msg: str) $\rightarrow str$

Remove known unit suffixes from a msg.

The units of some replies depend on the mode of the device, hence all known units are removed.

### property voltage

Get the measured source input voltage (float).

# 7.11 AJA International

This section contains specific documentation on the AJA International instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.11.1 AJA DCXS-750 or 1500 DC magnetron sputtering power supply

class pymeasure.instruments.aja.DCXS(adapter, name='AJA DCXS sputtering power supply', \*\*kwargs)

Bases: Instrument

AJA DCXS-750 or 1500 DC magnetron sputtering power supply with multiple outputs

Connection to the device is made through an RS232 serial connection. The communication settings are fixed in the device at 38400, one stopbit, no parity. The device's communication protocol uses single character commands and fixed length replies, both without any terminator.

#### **Parameters**

- adapter pyvisa resource name of the instrument or adapter instance
- name (string) The name of the instrument.
- kwargs Any valid key-word argument for Instrument

# property active\_gun

Control the active gun number.

#### **ask**(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return the read response.

#### **Parameters**

- **command** Command string to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- \*\*kwargs Keyword arguments passed to the read method.

#### Returns

String returned by the device without read\_termination.

# property current

Measure the output current in mA.

# property deposition\_time\_min

Control the minutes part of deposition time. Can be set only when 'enabled' is False.

# property deposition\_time\_sec

Control the seconds part of deposition time. Can be set only when 'enabled' is False.

### property enabled

Control the on/off state of the power supply

# property fault\_code

Get the error code from the power supply.

### property id

Get the power supply type identifier.

# property material

Control the material name of the sputter target.

### property power

Measure the actual output power in W.

#### property ramp\_time

Control the ramp time in seconds. Can be set only when 'enabled' is False.

# read(reply length=-1, \*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

#### property regulation\_mode

Control the regulation mode of the power supply.

## property remaining\_deposition\_time\_min

Get the minutes part of remaining deposition time.

# property remaining\_deposition\_time\_sec

Get the seconds part of remaining deposition time.

#### property setpoint

Control the setpoint value. Units are determined by regulation mode (power -> W, voltage -> V, current -> mA).

# property shutter\_delay

Control the shutter delay in seconds. Can be set only when 'enabled' is False.

7.11. AJA International 219

#### property shutter\_state

Get the status of the gun shutters. 0 for closed and 1 for open shutters.

# property software\_version

Get the software revision of the power supply firmware.

#### property voltage

Measure the output voltage in V.

# 7.12 Ametek

This section contains specific documentation on the Ametek instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.12.1 Ametek 7270 DSP Lockin Amplifier

class pymeasure.instruments.ametek.Ametek7270(adapter,  $name='Ametek\ DSP\ 7270'$ ,  $read\_termination= \x00'$ ,  $write\_termination= \x00'$ , \*\*kwargs)

Bases: SCPIUnknownMixin, Instrument

This is the class for the Ametek DSP 7270 lockin amplifier

In this instrument, some measurements are defined only for specific modes, called Reference modes, see set\_reference\_mode() and will raise errors if called incorrectly

#### property adc1

Get the input value of ADC1 in Volts (float).

## property adc2

Get the input value of ADC2 in Volts (float).

#### property adc3

Get the input value of ADC3 in Volts (float).

## property adc4

Get the input value of ADC4 in Volts (float).

ask(command, query\_delay=None)

Send a command and read the response, stripping white spaces.

Usually the properties use the *values()* method that adds a strip call, however several methods use directly the result from ask to be cast into some other types. It should therefore also add the strip here, as all responses end with a newline character.

# property auto\_gain

Control whether automatic gain is enabled (bool).

#### check\_set\_errors()

mandatory to be used for property setter

The Ametek protocol expect the default null character to be read to check the property has been correctly set. With default termination character set as Null character, this turns out as an empty string to be read.

### property dac1

Control the output value on DAC1 in Volts (float truncated to values from -10 to 10).

# property dac2

Control the output value on DAC2 in Volts (float truncated to values from -10 to 10).

### property dac3

Control the output value on DAC3 in Volts (float truncated to values from -10 to 10).

### property dac4

Control the output value on DAC4 in Volts (float truncated to values from -10 to 10).

### property frequency

Control the lock-in frequency in Hz (float, truncated\_range, values from 0 to 2.5e5).

## property harmonic

Control the reference harmonic mode (integer truncated to values from 1 to 127).

### property id

Get the instrument ID and firmware version

### property mag

Get magnitude in Volts (float).

### property phase

Control the reference harmonic phase in degrees (float, modular\_range, values from 0 to 360).

# property sensitivity

Control the sensitivity range in Volts (float truncated to values from 2 nV to 1 V).(dynamic)

# set\_channel\_A\_mode()

Sets instrument to channel A mode – assuming it is in voltage mode

## set\_current\_mode(low\_noise=False)

Sets instrument to current control mode with either low noise or high bandwidth

# set\_differential\_mode(lineFiltering=True)

Sets instrument to differential mode – assuming it is in voltage mode

# $set_reference_mode(mode: int = 0)$

Set the instrument in Single, Dual or harmonic mode.

#### **Parameters**

**mode** – the integer specifying the mode: 0 for Single, 1 for Dual harmonic, and 2 for Dual reference.

# set\_voltage\_mode()

Sets instrument to voltage control mode

#### shutdown()

Ensures the instrument in a safe state

#### property slope

Control the filter slope in dB/octave (integer truncated to values 6, 12, 18, 24).

# property theta

Get signal phase in degrees (float).

7.12. Ametek 221

#### property time\_constant

Control the time constant in seconds (float truncated to values from 10 microseconds to 100,000 seconds).

## property voltage

Control the voltage in Volts (float, truncated\_range, values from 0 to 5).

### property x

Get X value in Volts (float).

#### property x1

Get first harmonic X value in Volts (float).

## property x2

Get second harmonic X value in Volts (float).

# property xy

Get both X and Y values in Volts (float, tuple).

# property y

Get Y value in Volts (float).

# property y1

Get first harmonic Y value in Volts (float).

# property y2

Get second harmonic Y value in Volts (float).

# 7.13 AMI

This section contains specific documentation on the AMI instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.13.1 AMI 430 Power Supply

Bases: SCPIMixin, Instrument

Represents the AMI 430 Power supply and provides a high-level for interacting with the instrument.

```
magnet = AMI430("TCPIP::web.address.com::7180::SOCKET")
magnet.coilconst = 1.182
                                        # kGauss/A
magnet.voltage_limit = 2.2
                                        # Sets the voltage limit in V
magnet.target_current = 10
                                        # Sets the target current to 10 A
magnet.target_field = 1
                                        # Sets target field to 1 kGauss
magnet.ramp_rate_current = 0.0357
                                       # Sets the ramp rate in A/s
magnet.ramp_rate_field = 0.0422
                                       # Sets the ramp rate in kGauss/s
magnet.ramp
                                       # Initiates the ramping
magnet.pause
                                        # Pauses the ramping
```

(continues on next page)

(continued from previous page)

```
magnet.status  # Returns the status of the magnet

magnet.ramp_to_current(5)  # Ramps the current to 5 A

magnet.shutdown()  # Ramps the current to zero and disables_
output
```

## property coilconst

Control the coil constant in kGauss/A. (float)

# disable\_persistent\_switch()

Disables the persistent switch.

## enable\_persistent\_switch()

Enables the persistent switch.

# property field

Get the field in kGauss of the magnet.

### has\_persistent\_switch\_enabled()

Returns a boolean if the persistent switch is enabled.

# property magnet\_current

Get the current in Amps of the magnet.

#### property magnet\_status

Get the magnet status.

# pause()

Pauses the ramping of the magnetic field.

#### ramp()

Initiates the ramping of the magnetic field to set current/field with ramping rate previously set.

# property ramp\_rate\_current

Control the current ramping rate in A/s. (float)

# property ramp\_rate\_field

Control the field ramping rate in kGauss/s. (float)

## ramp\_to\_current(current, rate)

Heats up the persistent switch and ramps the current with set ramp rate.

# ramp\_to\_field(field, rate)

Heats up the persistent switch and ramps the current with set ramp rate.

### **shutdown**(ramp rate=0.0357)

Turns on the persistent switch, ramps down the current to zero, and turns off the persistent switch.

## property state

Get the field in kGauss of the magnet.

# property supply\_current

Get the current in Amps of the power supply.

7.13. AMI 223

```
property target_current
    Control the target current in A for the magnet. (float)

property target_field
    Control the target field in kGauss for the magnet. (float)

property voltage_limit
    Control the voltage limit for charging/discharging the magnet. (float)

wait_for_holding(should_stop=<function AMI430.<lambda>>, timeout=800, interval=0.1)
```

Initiates the ramping of the magnetic field to zero current/field with ramping rate previously set.

# 7.14 Anaheim Automation

zero()

This section contains specific documentation on the Anaheim Automation instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.14.1 DP-Series Step Motor Controller

The DPSeriesMotorController class implements a base driver class for Anaheim-Automation DP Series stepper motor controllers. There are many controllers sold in this series, all of which implement the same core command set. Some controllers, like the DPY50601, implement additional functionality that is not included in this driver. If these additional features are desired, they should be implemented in a subclass.

Bases: Instrument

Base class to interface with Anaheim Automation DP series stepper motor controllers.

This driver has been tested with the DPY50601 and DPE25601 motor controllers.

# property absolute\_position

Float property representing the value of the motor position measured in absolute units. Note that in DP series motor controller instrument manuals, 'absolute position' refers to the  $step\_position$  property rather than this property. Also note that use of this property relies on  $steps\_to\_absolute()$  and  $absolute\_to\_steps()$  being implemented in a subclass. In this way, the user can define the conversion from a motor step position into any desired absolute unit. Absolute units could be the position in meters of a linear stage or the angular position of a gimbal mount, etc. This property can be set.

## absolute\_to\_steps(pos)

Convert an absolute position to a number of steps to move. This must be implemented in subclasses.

#### **Parameters**

**pos** – Absolute position in the units determined by the subclassed *absolute\_to\_steps()* method.

### property address

Integer property representing the address that the motor controller uses for serial communications.

# property basespeed

Integer property that represents the motor controller's starting/homing speed. This property can be set.

### property busy

Query to see if the controller is currently moving a motor.

#### check\_errors()

Method to read the error codes register and log when an error is detected.

# Return error\_code

one byte with the error codes register contents

### property direction

A string property that represents the direction in which the stepper motor will rotate upon subsequent step commands. This property can be set. 'CW' corresponds to clockwise rotation and 'CCW' corresponds to counter-clockwise rotation.

## property encoder\_autocorrect

A boolean property to enable or disable the encoder auto correct function. This property can be set.

### property encoder\_delay

An integer property that represents the wait time in ms. after a move is finished before the encoder is read for a potential encoder auto-correct action to take place. This property can be set.

### property encoder\_enabled

A boolean property to represent whether an external encoder is connected and should be used to set the *step\_position* property.

### property encoder\_motor\_ratio

An integer property that represents the ratio of the number of encoder pulses per motor step. This property can be set.

#### property encoder\_retries

An integer property that represents the number of times the motor controller will try the encoder auto correct function before setting an error flag. This property can be set.

# property encoder\_window

An integer property that represents the allowable error in encoder pulses from the desired position before the encoder auto-correct function runs. This property can be set.

#### property error\_reg

Reads the current value of the error codes register.

# home(home\_mode)

Send command to the motor controller to 'home' the motor.

#### **Parameters**

**home\_mode** – **0** or **1** specifying which homing mode to run.

0 will perform a homing operation where the controller moves the motor until a soft limit is reached, then will ramp down to base speed and continue motion until a home limit is reached.

In mode 1, the controller will move the motor until a limit is reached, then will ramp down to base speed, change direction, and run until the limit is released.

#### property maxspeed

Integer property that represents the motor controller's maximum (running) speed. This property can be set.

#### move(direction)

Move the stepper motor continuously in the given direction until a stop command is sent or a limit switch is reached. This method corresponds to the 'slew' command in the DP series instrument manuals.

#### **Parameters**

**direction** – value to set on the direction property before moving the motor.

# reset\_position()

Reset position as counted by the motor controller and an externally connected encoder to 0.

# property step\_position

Integer property representing the value of the motor position measured in steps counted by the motor controller or, if <code>encoder\_enabled</code> is set, the steps counted by an externally connected encoder. Note that in the DP series motor controller instrument manuals, this property would be referred to as the 'absolute position' while this driver implements a conversion between steps and absolute units for the <code>absolute\_position</code> property. This property can be set.

### steps\_to\_absolute(steps)

Convert a position measured in steps to an absolute position.

#### **Parameters**

**steps** – Position in steps to be converted to an absolute position.

## stop()

Method that stops all motion on the motor controller.

## wait\_for\_completion(interval=0.5)

Block until the controller is not "busy" (i.e. block until the motor is no longer moving.)

### **Parameters**

**interval** – (float) seconds between queries to the "busy" flag.

#### Returns

None

# write(command)

Override the instrument base write method to add the motor controller's address to the command string.

#### **Parameters**

**command** – command string to be sent to the motor controller.

# 7.15 Anapico

This section contains specific documentation on the Anapico instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.15.1 Anapico APSIN12G Signal Generator

Bases: SCPIUnknownMixin, Instrument

Represents the Anapico APSIN12G Signal Generator with option 9K, HP and GPIB.

### property blanking

Control the blanking of output power when frequency is changed. ON makes the output to be blanked (off) while changing frequency.

# disable\_rf()

Disables the RF output.

## enable\_rf()

Enables the RF output.

## property frequency

Control the output frequency in Hz. (float)

# property power

Control the output power in dBm. (float)

# property reference\_output

Control the 10MHz reference output from the synth. (str)

# 7.16 Andeen Hagerling

This section contains specific documentation on the Andeen Hagerling instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.16.1 Andeen Hagerling AH2500A capacitance bridge

class pymeasure.instruments.andeenhagerling.AH2500A(adapter, name=None, timeout=3000, write\_termination= $\n'$ , read\_termination= $\n'$ , \*\*kwargs)

Bases: Instrument

Andeen Hagerling 2500A Precision Capacitance Bridge implementation

## property caplossvolt

Get the result of a single capacitance, loss measurement and return the values in units of pF and nS. The used measurement voltage is returned as third value.

# property config

Get the configuration.

#### trigger()

Triggers a new measurement without blocking and waiting for the return value.

# triggered\_caplossvolt()

reads the measurement value after the device was triggered by the trigger function.

### property vhighest

Control maximum RMS value of the used measurement voltage. Values of up to 15 V are allowed. The device will select the best suiting range below the given value.

# 7.16.2 Andeen Hagerling AH2700A capacitance bridge

Bases: AH2500A

Andeen Hagerling 2700A Precision Capacitance Bridge implementation

### property caplossvolt

Get the result of a single capacitance, loss measurement and return the values in units of pF and nS. The used measurement voltage is returned as third value.

#### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

## check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

# Returns

List of error entries.

# clear()

Clears the instrument status byte

#### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## property config

Get the configuration

# property frequency

Control test frequency used for the measurements. Allowed are values between 50 and 20000 Hz. The device selects the closest possible frequency to the given value.

#### property id

Get the instrument identification

# property next\_error

Get the next error of the instrument (tuple of code and message).

## property options

Get the device options installed.

## read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Resets the instrument.

## shutdown()

Brings the instrument to a safe and stable state

# property status

Get the status byte and Master Summary Status bit.

## trigger()

Triggers a new measurement without blocking and waiting for the return value.

## triggered\_caplossvolt()

reads the measurement value after the device was triggered by the trigger function.

# property vhighest

Control maximum RMS value of the used measurement voltage. Values of up to 15 V are allowed. The device will select the best suiting range below the given value.

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

### write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

### **Parameters**

- command command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- **command** Command to send.
- values The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.17 Anritsu

This section contains specific documentation on the Anritsu instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.17.1 Anritsu MG3692C Signal Generator

Bases: SCPIUnknownMixin, Instrument

Represents the Anritsu MG3692C Signal Generator

# disable()

Disables the signal output.

# enable()

Enables the signal output.

# property frequency

Control the output frequency in Hz. This property can be set. (float)

# property output

Control the signal output state. (bool)

# property power

Control the output power in dBm. (float))

### shutdown()

Shuts down the instrument, putting it in a safe state.

# 7.17.2 Anritsu MS9710C Optical Spectrum Analyzer

Bases: SCPIUnknownMixin, Instrument

Anritsu MS9710C Optical Spectrum Analyzer.

#### property analysis

Control Analysis Control

# property analysis\_result

Get anaysis result from current scan.

# property average\_point

Control number of averages to take on each point (2-1000), or OFF

#### property average\_sweep

Control number of averages to make on a sweep (2-1000) or OFF

## center\_at\_peak(\*\*kwargs)

Center the spectrum at the measured peak.

# property data\_memory\_a\_condition

Get the data condition of data memory register A. Starting wavelength, and a sampling point (11, 12, n).

# property data\_memory\_a\_size

Get the number of points sampled in data memory register A.

#### property data\_memory\_a\_values

Get the binary data from memory register A.

## property data\_memory\_b\_condition

Get the data condition of data memory register B. Starting wavelength, and a sampling point (11, 12, n).

#### property data\_memory\_b\_size

Get the number of points sampled in data memory register B.

# property data\_memory\_b\_values

Get the binary data from memory register B.

### property data\_memory\_select

Control Memory Data Select.

# property dip\_search

Control Dip Search Mode

#### property ese2

Control Extended Event Status Enable Register 2

# property esr2

Control Extended Event Status Register 2

#### property level\_lin

Control Level Linear Scale (/div)

# property level\_log

Control Level Log Scale (/div)

#### property level\_opt\_attn

Control Optical Attenuation Status (ON/OFF)

# property level\_scale

Get Current Level Scale

#### property measure\_mode

Get the current Measure Mode the OSA is in.

7.17. Anritsu 231

#### measure\_peak()

Measure the peak and return the trace marker.

### property peak\_search

Control Peak Search Mode

### read\_memory(slot='A')

Read the scan saved in a memory slot.

# property resolution

Control Resolution (nm)

# property resolution\_actual

Control Resolution Actual (ON/OFF)

## property resolution\_vbw

Control Video Bandwidth Resolution

# property sampling\_points

Control number of sampling points

# single\_sweep(\*\*kwargs)

Perform a single sweep and wait for completion.

## property trace\_marker

Control the trace marker with a wavelength. Returns the trace wavelength and power.

### property trace\_marker\_center

Set Trace Marker at Center. Set to 1 or True to initiate command

# wait(n=3, delay=1)

Query OPC Command and waits for appropriate response.

## $wait_for_sweep(n=20, delay=0.5)$

Wait for a sweep to stop.

This is performed by checking bit 1 of the ESR2.

# property wavelength\_center

Control Center Wavelength of Spectrum Scan in nm.

## property wavelength\_marker\_value

Control Wavelength Marker Value (wavelength or freq.?)

#### property wavelength\_span

Control Wavelength Span of Spectrum Scan in nm.

# property wavelength\_start

Control Wavelength Start of Spectrum Scan in nm.

### property wavelength\_stop

Control Wavelength Stop of Spectrum Scan in nm.

# property wavelength\_value\_in

Control Wavelength value in Vacuum or Air

# property wavelengths

Get a numpy array of the current wavelengths of scans.

# 7.17.3 Anritsu MS9740A Optical Spectrum Analyzer

Bases: AnritsuMS9710C

Anritsu MS9740A Optical Spectrum Analyzer.

#### property average\_sweep

Control number of averages to make on a sweep (1-1000), with 1 being a single (non-averaged) sweep

# property data\_memory\_select

Control Memory Data Select.

repeat\_sweep(n=20, delay=0.5)

Perform a single sweep and wait for completion.

### property resolution

Control Resolution (nm)

### property resolution\_vbw

Control Video Bandwidth Resolution

# property sampling\_points

Control number of sampling points

# 7.17.4 Anritsu MS2090A Handheld Spectrum Analyzer

Bases: SCPIMixin, Instrument

Anritsu MS2090A Handheld Spectrum Analyzer.

### abort()

Initiate a sweep/measurement.

#### property active\_state

The "set" state indicates that the instrument is used by someone.

# property external\_current

This command queries the actual bias current in A

## property fetch\_control

Returns the Control Channel measurement in json format.

# property fetch\_density

Returns the most recent channel density measurement

## property fetch\_eirpower

Returns the current EIRP, Max EIRP, Horizontal EIRP, Vertical and Sum EIRP results in dBm.

# property fetch\_eirpower\_data

This command returns the current EIRP measurement result in dBm.

#### property fetch\_eirpower\_max

This command returns the Max EIRP measurement result in dBm.

7.17. Anritsu 233

### property fetch\_emf

Return the current EMF measurement data. JSON format.

# property fetch\_emf\_meter

Return the live EMF measurement data. JSON format.

## property fetch\_emf\_meter\_sample

Return the EMF measurement data for a specified sample number. JSON format.

### property fetch\_interference\_power

Fetch Interference Finder Integrated Power.

### property fetch\_mimo\_antenas

Returns the sync power measurement in json format.

## property fetch\_ocupied\_bw

Returns the different set of measurement information depending on the suffix.

# property fetch\_ota\_mapping

Returns the most recent Coverage Mapping measurement result.

# property fetch\_pan

Return the current Pulse Analyzer measurement data. JSON format

## property fetch\_pbch\_constellation

Get the latest Physical Broadcast Channel constellation hitmap

# property fetch\_pci

Returns PCI measurements

# property fetch\_pdsch

Returns the Data Channel Measurements in JSON format.

## property fetch\_pdsch\_constellation

Get the latest Physical Downlink Shared Channel constellation

# property fetch\_peak

Returns a pair of peak amplitude in current sweep.

# property fetch\_power

Returns the most recent channel power measurement.

#### property fetch\_rrm

Returns the Radio Resource Management in JSON format.

## property fetch\_scan

Returns the cell scanner measurements in JSON format

### property fetch\_semask

This command returns the current Spectral Emission Mask measurement result.

#### property fetch\_ssb

Returns the SSB measurement

# property fetch\_sync\_evm

Returns the Sync EVM measurement in JSON format.

### property fetch\_sync\_power

Returns the sync power measurements in JSON format

# property fetch\_tae

Returns the Time Alignment Error in JSON format.

#### property frequency\_center

Sets the center frequency in Hz

# property frequency\_offset

Sets the frequency offset in Hz

### property frequency\_span

Sets the frequency span in Hz

## property frequency\_span\_full

Sets the frequency span to full span

# property frequency\_span\_last

Sets the frequency span to the previous span value.

# property frequency\_start

Sets the start frequency in Hz

## property frequency\_step

Set or query the step size to gradually increase or decrease frequency values in Hz

# property frequency\_stop

Sets the start frequency in Hz

# property gps

Returns the timestamp, latitude, and longitude of the device.

## property gps\_all

Returns the fix timestamp, latitude, longitude, altitude and information on the sat used.

# property gps\_full

Returns the timestamp, latitude, longitude, altitude, and satellite count of the device.

# property gps\_last

Returns the timestamp, latitude, longitude, and altitude of the last fixed GPS result.

## init\_all\_sweep()

Initiate all sweep/measurement.

# property init\_continuous

Specified whether the sweep/measurement is triggered continuously

### property init\_spa\_self

Perform a self-test and return the results.

# init\_sweep()

Initiate a sweep/measurement.

# property meas\_acpower

Sets the active measurement to adjacent channel power ratio, sets the default measurement parameters, triggers a new measurement and returns the main channel power, lower adjacent, upper adjacent, lower alternate and upper alternate channel power results.

7.17. Anritsu 235

### property meas\_emf\_meter\_clear\_all

Clear the EMF measurement data of all samples. Sampling state will be turned off if it was on.

# property meas\_emf\_meter\_clear\_sample

Clear the EMF measurement data for a specified sample number. Sampling state will be turned off if the specified sample is currently active.

# property meas\_emf\_meter\_sample

Start or Stop applying the measurement results to the currently selected sample

### property meas\_int\_power

Sets the active measurement to interference finder, sets the default measurement parameters, triggers a new measurement and returns integrated power as the result. It is a combination of the commands :CONFigure:INTerference; :READ:INTerference:POWer?

# property meas\_iq\_capture

This set command is used to start the IQ capture measurement.

## property meas\_iq\_capture\_fail

Sets or queries whether the instrument will automatically save an IQ capture when losing sync

### property meas\_ota\_mapp

Sets the active measurement to OTA Coverage Mapping, sets the default measurement parameters, triggers a new measurement, and returns the measured values.

# property meas\_ota\_run

Turn on/off OTA Coverage Mapping Data Collection. The instrument must be in Coverage Mapping measurement for the command to be effective

## property meas\_power

Sets the active measurement to channel power, sets the default measurement parameters, triggers a new measurement and returns channel power as the result. It is a combination of the commands :CONFigure:CHPower; :READ:CHPower:CHPower?

# property meas\_power\_all

Sets the active measurement to channel power, sets the default measurement parameters, triggers a new measurement and returns the channel power and channel power density results. It is a combination of the commands: CONFigure: CHPower; :READ: CHPower?

# property power\_density

Sets the active measurement to channel power, sets the default measurement parameters, triggers a new measurement and returns channel power density as the result. It is a combination of the commands: CON-Figure: CHPower; :READ: CHPower: DENSity?

#### property preamp

Sets the state of the preamp. Note that this may cause a change in the reference level and/or attenuation.

# property sense\_mode

Set the operational mode of the Spa app.

#### property view\_sense\_modes

Returns a list of available modes for the Spa application. The response is a comma-separated list of mode names. See command [:SENSe]:MODE for the mode name specification.

# 7.17.5 Anritsu MS464xB Vector Network Analyzer

Bases: AnritsuMS464xB

A class representing the Anritsu MS4642B Vector Network Analyzer (VNA).

This VNA has a frequency range from 10 MHz to 20 GHz and is part of the *AnritsuMS464xB* family of instruments; for documentation, for documentation refer to this base class.

Bases: AnritsuMS464xB

A class representing the Anritsu MS4644B Vector Network Analyzer (VNA).

This VNA has a frequency range from 10 MHz to 40 GHz and is part of the *AnritsuMS464xB* family of instruments; for documentation, for documentation refer to this base class.

Bases: AnritsuMS464xB

A class representing the Anritsu MS4645B Vector Network Analyzer (VNA).

This VNA has a frequency range from 10 MHz to 50 GHz and is part of the *AnritsuMS464xB* family of instruments; for documentation, for documentation refer to this base class.

Bases: AnritsuMS464xB

A class representing the Anritsu MS4647B Vector Network Analyzer (VNA).

This VNA has a frequency range from 10 MHz to 70 GHz and is part of the *AnritsuMS464xB* family of instruments; for documentation, for documentation refer to this base class.

Bases: SCPIUnknownMixin, Instrument

A class representing the Anritsu MS464xB Vector Network Analyzer (VNA) series.

This family consists of the MS4642B, MS4644B, MS4645B, and MS4647B, which are represented in their respective classes (*AnritsuMS4642B*, *AnritsuMS4644B*, *AnritsuMS4645B*, *AnritsuMS4647B*), that only differ in the available frequency range.

They can contain up to 16 instances of MeasurementChannel (depending on the configuration of the instrument), that are accessible via the *channels* dict or directly via  $ch_{-}$  + the channel number.

7.17. Anritsu 237

#### **Parameters**

- active\_channels (int (1-16) or str ("auto")) defines the number of active channels (default=16); if active\_channels is "auto", the instrument will be queried for the number of active channels.
- installed\_ports (int (1-4) or str ("auto")) defines the number of installed ports (default=4); if "auto" is provided, the instrument will be queried for the number of ports
- traces\_per\_channel (int (1-16) or str ("auto") or None) defines the number of traces that is assumed for each channel (between 1 and 16); if not provided, the maximum number is assumed; "auto" is provided, the instrument will be queried for the number of traces of each channel.

### property active\_channel

Control the active channel.

# property bandwidth\_enhancer\_enabled

Control the state of the IF bandwidth enhancer.

# property binary\_data\_byte\_order

Control the binary numeric I/O data byte order.

valid values are:

value	description
NORM	The most significant byte (MSB) is first
SWAP	The least significant byte (LSB) is first

### check\_errors()

Read all errors from the instrument.

# Returns

list of error entries

# copy\_data\_file(from\_filename, to\_filename)

Copy a file on the VNA HDD.

#### **Parameters**

- **from\_filename** (*str*) full filename including pat
- to\_filename (str) full filename including path

# create\_directory(dir\_name)

Create a directory on the VNA HDD.

#### **Parameters**

dir\_name (str) - directory name

# property data\_drawing\_enabled

Control whether data drawing is enabled (True) or not (False).

# property datablock\_header\_format

Control the way the arbitrary block header for output data is formed.

Valid values are:

value	description
0	A block header with arbitrary length will be sent.
1	The block header will have a fixed length of 11 characters.
2	No block header will be sent. Not IEEE 488.2 compliant.

# property datablock\_numeric\_format

Control format for numeric I/O data representation.

Valid values are:

value	description
ASCII	An ASCII number of 20 or 21 characters long with floating point notation.
8byte	8 bytes of binary floating point number representation limited to 64 bits.
4byte	4 bytes of floating point number representation.

# property datafile\_frequency\_unit

Control the frequency unit displayed in a SNP data file.

Valid values are HZ, KHZ, MHZ, GHZ.

# property datafile\_include\_heading

Control whether a heading is included in the data files.

# property datafile\_parameter\_format

Control the parameter format displayed in an SNP data file.

Valid values are:

value	description
LINPH	Linear and Phase.
LOGPH	Log and Phase.
REIM	Real and Imaginary Numbers.

## delete\_data\_file(filename)

Delete a file on the VNA HDD.

#### **Parameters**

**filename** (str) – full filename including path

## delete\_directory(dir\_name)

Delete a directory on the VNA HDD.

### **Parameters**

 $dir_name(str) - directory name$ 

## property display\_layout

Control the channel display layout in a Row-by-Column format.

Valid values are: R1C1, R1C2, R2C1, R1C3, R3C1, R2C2C1, R2C1C2, C2R2R1, C2R1R2, R1C4, R4C1, R2C2, R2C3, R3C2, R2C4, R4C2, R3C3, R5C2, R2C5, R4C3, R3C4, R4C4. The number following the R indicates the number of rows, following the C the number of columns; e.g. R2C2 results in a 2-by-2 layout. The options that contain two C's or R's result in asymmetric layouts; e.g. R2C1C2 results in a layout with 1 channel on top and two channels side-by-side on the bottom row.

7.17. Anritsu 239

### property event\_status\_enable\_bits

Control the Standard Event Status Enable Register bits.

The register can be queried using the *query\_event\_status\_register()* method. Valid values are between 0 and 255. Refer to the instrument manual for an explanation of the bits.

## property external\_trigger\_delay

Control the delay time of the external trigger in seconds.

Valid values are between 0 [s] and 10 [s] in steps of 1e-9 [s] (i.e. 1 ns).

## property external\_trigger\_edge

Control the edge type of the external trigger.

Valid values are POS (for positive or leading edge) or NEG (for negative or trailing edge).

## property external\_trigger\_handshake

Control status of the external trigger handshake.

# property external\_trigger\_type

Control the type of trigger that will be associated with the external trigger.

Valid values are POIN (for point), SWE (for sweep), CHAN (for channel), and ALL.

### property hold\_function\_all\_channels

Control the hold function of all channels.

Valid values are:

value	description
CONT	Perform continuous sweeps on all channels
HOLD	Hold the sweep on all channels
SING	Perform a single sweep and then hold all channels

# load\_data\_file(filename)

Load a data file from the VNA HDD into the VNA memory.

#### **Parameters**

**filename** (str) – full filename including path

## load\_data\_file\_to\_memory(filename)

Load a data file to a memory trace.

# **Parameters**

**filename** (str) – full filename including path

## property manual\_trigger\_type

Control the type of trigger that will be associated with the manual trigger.

Valid values are POIN (for point), SWE (for sweep), CHAN (for channel), and ALL.

# property max\_number\_of\_points

Control the maximum number of points the instrument can measure in a sweep.

Note that when this value is changed, the instrument will be rebooted. Valid values are 25000 and 100000. When 25000 points is selected, the instrument supports 16 channels with 16 traces each; when 100000 is selected, the instrument supports 1 channel with 16 traces.

# property number\_of\_channels

Control the number of displayed (and therefore accessible) channels.

When the system is in 25000 points mode, the number of channels can be 1, 2, 3, 4, 6, 8, 9, 10, 12, or 16; when the system is in 100000 points mode, the system only supports 1 channel. If a value is provided that is not valid in the present mode, the instrument is set to the next higher channel number.

# property number\_of\_ports

Get the number of instrument test ports.

### query\_event\_status\_register()

Query the value of the Standard Event Status Register.

Note that querying this value, clears the register. Refer to the instrument manual for an explanation of the returned value.

# read\_datafile(channel, sweep\_points, datafile\_freq, datafile\_par, filename)

Read a data file from the VNA.

#### **Parameters**

- channel (int) Channel Index
- **sweep\_points** (*int*) number of sweep point as an integer
- datafile\_freq (DataFileFrequencyUnits) Data file frequency unit
- datafile\_par (DataFileParameter) Data file parameter format
- **filename** (*str*) full path of the file to be saved

### property remote\_trigger\_type

Control the type of trigger that will be associated with the remote trigger.

Valid values are POIN (for point), SWE (for sweep), CHAN (for channel), and ALL.

## return\_to\_local()

Returns the instrument to local operation.

# property service\_request\_enable\_bits

Control the Service Request Enable Register bits.

Valid values are between 0 and 255; setting 0 performs a register reset. Refer to the instrument manual for an explanation of the bits.

#### store\_image(filename)

Capture a screenshot to the file specified.

### **Parameters**

**filename** (str) – full filename including path

# trigger()

Trigger a continuous sweep from the remote interface.

## trigger\_continuous()

Trigger a continuous sweep from the remote interface.

# trigger\_single()

Trigger a single sweep with synchronization from the remote interface.

7.17. Anritsu 241

#### property trigger\_source

Control the source of the sweep/measurement triggering.

Valid values are:

value	description
AUTO	Automatic triggering
MAN	Manual triggering
EXTT	Triggering from rear panel BNC via the GPIB parser
EXT	External triggering port
REM	Remote triggering

# update\_channels(number\_of\_channels=None, \*\*kwargs)

Create or remove channels to be correct with the actual number of channels.

#### **Parameters**

**number\_of\_channels** (*int*) – optional, if given, defines the desired number of channels.

class pymeasure.instruments.anritsu.anritsuMS464xB.MeasurementChannel(\*args,

frequency\_range=None,
traces=None, \*\*kwargs)

Bases: Channel

Represents a channel of Anritsu MS464xB VNA.

Contains 4 instances of *Port* (accessible via the *ports* dict or directly  $pt_{-}$  + the port number) and up to 16 instances of *Trace* (accessible via the *traces* dict or directly  $tr_{-}$  + the trace number).

## **Parameters**

- **frequency\_range** (*list of floats*) defines the number of installed ports (default=4).
- **traces** (*int* (1-16) or str ("auto") or None) defines the number of traces that is assumed for the channel (between 1 and 16); if not provided, the maximum number is assumed; "auto" is provided, the instrument will be queried for the number of traces.

#### activate()

Set the indicated channel as the active channel.

# property active\_trace

Set the active trace on the indicated channel.

### property application\_type

Control the application type of the specified channel.

Valid values are TRAN (for transmission/reflection), NFIG (for noise figure measurement), PULS (for PulseView).

# property average\_count

Control the averaging count for the indicated channel.

The channel must be turned on. Valid values are between 1 and 1024.

### property average\_sweep\_count

Get the averaging sweep count for the indicated channel.

#### property average\_type

Control the averaging type to for the indicated channel.

Valid values are POIN (point-by-point) or SWE (sweep-by-sweep)

### property averaging\_enabled

Control whether the averaging is turned on for the indicated channel.

### property bandwidth

Control the IF bandwidth for the indicated channel.

Valid values are between 1 [Hz] and 1E6 [Hz] (i.e. 1 MHz). The system will automatically select the closest IF bandwidth from the available options (1, 3, 10 ... 1E5, 3E5, 1E6).

# property calibration\_enabled

Control whether the RF correction (calibration) is enabled for indicated channel.

# check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

# clear\_average\_count()

Clear and restart the averaging sweep count of the indicated channel.

### property cw mode enabled

Control the state of the CW sweep mode of the indicated channel.

# property cw\_number\_of\_points

Control the CW sweep mode number of points of the indicated channel.

Valid values are between 1 and 25000 or 100000 depending on the maximum points setting.

## property display\_layout

Control the trace display layout in a Row-by-Column format for the indicated channel.

Valid values are: R1C1, R1C2, R2C1, R1C3, R3C1, R2C2C1, R2C1C2, C2R2R1, C2R1R2, R1C4, R4C1, R2C2, R2C3, R3C2, R2C4, R4C2, R3C3, R5C2, R2C5, R4C3, R3C4, R4C4. The number following the R indicates the number of rows, following the C the number of columns; e.g. R2C2 results in a 2-by-2 layout. The options that contain two C's or R's result in asymmetric layouts; e.g. R2C1C2 results in a layout with 1 trace on top and two traces side-by-side on the bottom row.

#### property frequency\_CW

Control the CW frequency of the indicated channel in hertz.

Valid values are between 1E7 [Hz] (i.e. 10 MHz) and 4E10 [Hz] (i.e. 40 GHz). (dynamic)

# property frequency\_center

Control the center value of the sweep range of the indicated channel in hertz.

Valid values are between 1E7 [Hz] (i.e. 10 MHz) and 4E10 [Hz] (i.e. 40 GHz). (dynamic)

#### property frequency\_span

Control the span value of the sweep range of the indicated channel in hertz.

Valid values are between 2 [Hz] and 4E10 [Hz] (i.e. 40 GHz). (dynamic)

7.17. Anritsu 243

### property frequency\_start

Control the start value of the sweep range of the indicated channel in hertz.

Valid values are between 1E7 [Hz] (i.e. 10 MHz) and 4E10 [Hz] (i.e. 40 GHz). (dynamic)

# property frequency\_stop

Control the stop value of the sweep range of the indicated channel in hertz.

Valid values are between 1E7 [Hz] (i.e. 10 MHz) and 4E10 [Hz] (i.e. 40 GHz). (dynamic)

# property hold\_function

Control the hold function of the specified channel.

valid values are:

value	description
CONT	Perform continuous sweeps on all channels
HOLD	Hold the sweep on all channels
SING	Perform a single sweep and then hold all channels

## property number\_of\_points

Control the number of measurement points in a frequency sweep of the indicated channel.

Valid values are between 1 and 25000 or 100000 depending on the maximum points setting.

# property number\_of\_traces

Control the number of traces on the specified channel

Valid values are between 1 and 16.

## property sweep\_mode

Control the sweep mode for Spectrum Analysis on the indicated channel.

Valid options are VNA (for a VNA-like mode where the instrument will only measure at points in the frequency list) or CLAS (for a classical mode, where the instrument will scan all frequencies in the range).

# property sweep\_time

Control the sweep time of the indicated channel.

Valid values are between 2 and 100000.

### property sweep\_type

Control the sweep type of the indicated channel.

Valid options are:

value	description	
LIN	Frequency-based linear sweep	
LOG	Frequency-based logarithmic sweep	
FSEGM	Segment-based sweep with frequency-based segments	
ISEGM	Index-based sweep with frequency-based segments	
POW	Power-based sweep with either a CW frequency or swept-frequency	
MFGC	Multiple frequency gain compression	

# update\_frequency\_range(frequency\_range)

Update the values-attribute of the frequency-related dynamic properties.

### **Parameters**

**frequency\_range** (*list*) – the frequency range that the instrument is capable of.

## update\_traces(number\_of\_traces=None)

Create or remove traces to be correct with the actual number of traces.

#### **Parameters**

**number\_of\_traces** (*int*) – optional, if given defines the desired number of traces.

class pymeasure.instruments.anritsu.anritsuMS464xB.Trace(parent, id)

Bases: Channel

Represents a trace within a MeasurementChannel of the Anritsu MS464xB VNA.

#### activate()

Set the indicated trace as the active one.

#### property measurement\_parameter

Control the measurement parameter of the indicated trace.

Valid values are any S-parameter (e.g. S11, S12, S41) for 4 ports, or one of the following:

value	description	
Sxx	S-parameters (1-4 for both x)	
MIX	Response Mixed Mode	
NFIG	Noise Figure trace response (only with option 41 or 48)	
NPOW	Noise Power trace response (only with option 41 or 48)	
NTEMP	Noise Temperature trace response (only with option 41 or 48)	
AGA	Noise Figure Available Gain trace response (only with option 48)	
IGA	Noise Figure Insertion Gain trace response (only with option 48)	

class pymeasure.instruments.anritsu.anritsuMS464xB.Port(parent, id)

Bases: Channel

Represents a port within a MeasurementChannel of the Anritsu MS464xB VNA.

#### property power\_level

Control the power level (in dBm) of the indicated port on the indicated channel.

# 7.18 Attocube

This section contains specific documentation on the Attocube instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

7.18. Attocube 245

# 7.18.1 Attocube ANC300 Motion Controller

Bases: Instrument

Attocube ANC300 Piezo stage controller with several axes

#### **Parameters**

- adapter The VISA resource name of the controller (e.g. "TCPIP::<address>::<port>::SOCKET") or a created Adapter. The instruments default communication port is 7230.
- axisnames a list of axis names which will be used to create properties with these names
- passwd password for the attocube standard console
- query\_delay default delay between sending and reading in s (default 0.05)
- **host** host address of the instrument (e.g. 169.254.0.1)

Deprecated since version 0.11.2: The 'host' argument is deprecated. Use 'adapter' argument instead.

• kwargs – Any valid key-word argument for VISAAdapter

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

#### Returns

List of error entries.

#### property controllerBoardVersion

Get the serial number of the controller board.

### ground\_all()

Grounds all axis of the controller.

# handle\_deprecated\_host\_arg(adapter, kwargs)

This function formats user input to the \_\_init\_\_ function to be compatible with the current definition of the \_\_init\_\_ function. This is used to support outdated (deprecated) code. and separated out to make it easier to remove in the future. To whoever removes this: This function should be removed and the *adapter* argument in the \_\_init\_\_ method should be made non-optional.

#### Parameters

**kwargs** (dict) – keyword arguments passed to the <u>\_\_init\_\_</u> function, including the deprecated *host* argument.

#### Return str

resource string for the VISAAdapter

# read()

Read after setting a value.

# stop\_all()

Stop all movements of the axis.

### property version

Get the version number and instrument identification.

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None means query\_delay.

### **class** pymeasure.instruments.attocube.anc300.**Axis**(parent, id)

Bases: Channel

Represents a single open loop axis of the Attocube ANC350

#### **Parameters**

- axis axis identifier, integer from 1 to 7
- controller ANC300Controller instance used for the communication

#### property capacity

Measure the saved capacity value in nF of the axis.

### property frequency

Control the frequency of the stepping motion in Hertz from 1 to 10000 Hz.

#### insert\_id(command)

Insert the channel id in a command replacing *placeholder*.

Add axis id to a command string at the correct position after the initial command, but before a potential value.

# measure\_capacity()

Obtains a new measurement of the capacity. The mode of the axis returns to 'gnd' after the measurement.

## **Returns capacity**

the freshly measured capacity in nF.

# property mode

Control axis mode. This can be 'gnd', 'inp', 'cap', 'stp', 'off', 'stp+', 'stp-'. Available modes depend on the actual axis model.

#### **move**(*steps*, *gnd=True*)

Move 'steps' steps in the direction given by the sign of the argument. This method will change the mode of the axis automatically and ground the axis on the end if 'gnd' is True. The method is blocking and returns only when the movement is finished.

#### **Parameters**

- **steps** finite integer value of steps to be performed. A positive sign corresponds to upwards steps, a negative sign to downwards steps.
- gnd bool, flag to decide if the axis should be grounded after completion of the movement

# move\_raw(steps)

Move 'steps' steps in the direction given by the sign of the argument. This method assumes the mode of the axis is set to 'stp' and it is non-blocking, i.e. it will return immediately after sending the command.

#### **Parameters**

**steps** – integer value of steps to be performed. A positive sign corresponds to upwards steps, a negative sign to downwards steps. The values of +/-inf trigger a continuous movement. The axis can be halted by the stop method.

7.18. Attocube 247

#### property offset\_voltage

Control offset voltage in Volts from 0 to 150 V.

#### property output\_voltage

Measure the output voltage in volts.

## property pattern\_down

Control step down pattern of the piezo drive. 256 values ranging from 0 to 255 representing the sequence of output voltages within one step of the piezo drive. This property can be set, the set value needs to be an array with 256 integer values.

#### property pattern\_up

Control step up pattern of the piezo drive. 256 values ranging from 0 to 255 representing the sequence of output voltages within one step of the piezo drive. This property can be set, the set value needs to be an array with 256 integer values.

### property serial\_nr

Get the serial number of the axis.

### property stepd

Set the steps downwards for N steps. Mode must be 'stp' and N must be positive. 0 causes a continuous movement until stop is called.

Deprecated since version 0.13.0: Use meth: move\_raw instead.

### property stepu

Set the steps upwards for N steps. Mode must be 'stp' and N must be positive. 0 causes a continuous movement until stop is called.

Deprecated since version 0.13.0: Use meth: move\_raw instead.

## stop()

Stop any motion of the axis

#### property voltage

Control the amplitude of the stepping voltage in volts from 0 to 150 V.

# 7.19 BK Precision

This section contains specific documentation on the BK Precision instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.19.1 BK Precision 9130B DC Power Supply

Bases: SCPIUnknownMixin, Instrument

Represents the BK Precision 9130B DC Power Supply interface for interacting with the instrument.

#### property channel

Control which channel is selected. Can only take values [1, 2, 3]. (int)

# property current

Control the current of the selected channel. (float)

#### property source\_enabled

Control whether the source is enabled. (bool)

## property voltage

Control voltage of the selected channel. (float)

# 7.20 Danfysik

This section contains specific documentation on the Danfysik instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.20.1 Danfysik 8500 Power Supply

Bases: Instrument

Represents the Danfysik 8500 Electromanget Current Supply and provides a high-level interface for interacting with the instrument

To allow user access to the Prolific Technology PL2303 Serial port adapter in Linux, create the file: /etc/udev/rules.d/50-danfysik.rules, with contents:

```
SUBSYSTEMS=="usb",ATTRS{idVendor}=="067b",ATTRS{idProduct}=="2303",MODE="0666",

SYMLINK+="danfysik"
```

Then reload the udev rules with:

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

The device will be accessible through the port /dev/danfysik.

```
add_ramp_step(current)
```

Adds a current step to the ramp set.

#### **Parameters**

**current** – A current in Amps

clear\_ramp\_set()

Clears the ramp set.

clear\_sequence(stack)

Clears the sequence by the stack number.

**Parameters** 

**stack** – A stack number between 0-15

property current

Control the actual current in Amps. This property can be set through *current\_ppm*.

property current\_ppm

Control the current in parts per million..

7.20. Danfysik 249

#### property current\_setpoint

Get the setpoint for the current, which can deviate from the actual current (*current*) while the supply is in the process of setting the value.

#### disable()

Disables the flow of current.

### enable()

Enables the flow of current.

# property id

Get the idenfitication information.

#### is\_current\_stable()

Returns True if the current is within 0.02 A of the setpoint value.

#### is\_enabled()

Returns True if the current supply is enabled.

### is\_ready()

Returns True if the instrument is in the ready state.

#### is\_sequence\_running(stack)

Returns True if a sequence is running with a given stack number

#### Parameters

**stack** – A stack number between 0-15

#### local()

Sets the instrument in local mode, where the front panel can be used.

# property polarity

Control the polarity of the current supply, being either -1 or 1. This property can be set by supplying one of these values.

#### ramp\_to\_current(current, points, delay\_time=1)

Executes *set\_ramp\_to\_current()* and starts the ramp.

# read()

Read the device and raise exceptions if errors are reported by the instrument.

### Returns

String ASCII response of the instrument

#### Raises

An Exception if the Danfysik raises an error

# remote()

Sets the instrument in remote mode, where the front panel is disabled.

#### reset\_interlocks()

Resets the instrument interlocks.

#### set\_ramp\_delay(time)

Sets the ramp delay time in seconds.

#### **Parameters**

**time** – The time delay time in seconds

#### set\_ramp\_to\_current(current, points, delay\_time=1)

Sets up a linear ramp from the initial current to a different current, with a number of points, and delay time.

#### **Parameters**

- **current** The final current in Amps
- points The number of linear points to traverse
- delay\_time A delay time in seconds

### set\_sequence(stack, currents, times, multiplier=999999)

Sets up an arbitrary ramp profile with a list of currents (Amps) and a list of interval times (seconds) on the specified stack number (0-15)

### property slew\_rate

Get the slew rate of the current sweep.

#### start\_ramp()

Starts the current ramp.

### start\_sequence(stack)

Starts a sequence by the stack number.

#### **Parameters**

**stack** – A stack number between 0-15

## property status

Get a list of human-readable strings that contain the instrument status information, based on status\_hex.

## property status\_hex

Get the status in hexadecimal. This value is parsed in *status* into a human-readable list.

# stop\_ramp()

Stops the current ramp.

## stop\_sequence()

Stops the currently running sequence.

# sync\_sequence(stack, delay=0)

Arms the ramp sequence to be triggered by a hardware input to pin P33 1&2 (10 to 24 V) or a TS command. If a delay is provided, the sequence will start after the delay.

#### **Parameters**

- **stack** A stack number between 0-15
- delay A delay time in seconds

# wait\_for\_current(has\_aborted=<function Danfysik8500.<lambda>>, delay=0.01)

Blocks the process until the current has stabilized. A provided function has\_aborted can be supplied, which is checked after each delay time (in seconds) in addition to the stability check. This allows an abort feature to be integrated.

#### **Parameters**

- has\_aborted A function that returns True if the process should stop waiting
- **delay** The delay time in seconds between each check for stability

7.20. Danfysik 251

```
wait_for_ready(has_aborted=<function Danfysik8500.<lambda>>, delay=0.01)
```

Blocks the process until the instrument is ready. A provided function has\_aborted can be supplied, which is checked after each delay time (in seconds) in addition to the readiness check. This allows an abort feature to be integrated.

#### **Parameters**

- has\_aborted A function that returns True if the process should stop waiting
- **delay** The delay time in seconds between each check for readiness

# 7.21 Delta Elektronika

This section contains specific documentation on the Delta Elektronika instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.21.1 Delta Elektronica SM7045D Power source

Bases: SCPIUnknownMixin, Instrument

This is the class for the SM 70-45 D power supply.

```
source = SM7045D("GPIB::8")

source.ramp_to_zero(1)  # Set output to 0 before enabling
source.enable()  # Enables the output
source.current = 1  # Sets a current of 1 Amps
```

# property current

A floating point property that represents the output current of the power supply in Amps. This property can be set.

## disable()

Enables remote shutdown, hence input will be disabled.

# enable()

Disable remote shutdown, hence output will be enabled.

## property max\_current

A floating point property that represents the maximum output current of the power supply in Amps. This property can be set.

#### property max\_voltage

A floating point property that represents the maximum output voltage of the power supply in Volts. This property can be set.

### property measure\_current

Measures the actual output current of the power supply in Amps.

### property measure\_voltage

Measures the actual output voltage of the power supply in Volts.

#### ramp\_to\_current(target\_current, current\_step=0.1)

Gradually increase/decrease current to target current.

# **Parameters**

- target\_current Float that sets the target current (in A)
- **current\_step** Optional float that sets the current steps / ramp rate (in A/s)

```
ramp_to_zero(current_step=0.1)
```

Gradually decrease the current to zero.

#### **Parameters**

**current\_step** – Optional float that sets the current steps / ramp rate (in A/s)

### property rsd

Check whether remote shutdown is enabled/disabled and thus if the output of the power supply is disabled/enabled.

#### shutdown()

Set the current to 0 A and disable the output of the power source.

### property voltage

A floating point property that represents the output voltage setting of the power supply in Volts. This property can be set.

# 7.22 Edwards

This section contains specific documentation on the Edwards instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.22.1 Edwards nxds vacuum pump

class pymeasure.instruments.edwards.Nxds(adapter, name='Edwards NXDS Vacuum Pump', \*\*kwargs)

Bases: Instrument

Represents the Edwards nXDS (10i) Vacuum Pump and provides a low-level interaction with the instrument. This could potentially work with Edwards pump that has a RS232 interface. This instrument is constructed to only start and stop pump.

### property enable

Set the pump enabled state with default settings.

# 7.23 EURO TEST

This section contains specific documentation on the EURO TEST instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

7.22. Edwards 253

# 7.23.1 Euro Test HPP120256 High Voltage Power Supply

Bases: Instrument

Represents the Euro Test High Voltage DC Source model HPP-120-256 and provides a high-level interface for interacting with the instrument using the Euro Test command set (Not SCPI command set).

```
hpp120256 = EurotestHPP120256("GPIB0::20::INSTR")
print(hpp120256.id)
print(hpp120256.lam_status)
print(hpp120256.status)
hpp120256.ramp_to_zero(100.0)
hpp120256.voltage_ramp = 50.0 \# V/s
hpp120256.current_limit = 2.0 # mA
inst.kill_enabled = True # Enable over-current protection
time.sleep(1.0) # Give time to enable kill
inst.output_enabled = True
time.sleep(1.0) # Give time to output on
abs_output_voltage_error = 0.02 # kV
hpp120256.wait_for_output_voltage_reached(abs_output_voltage_error, 1.0, 40.0)
# Here voltage HV output should be at 0.0 kV
print("Setting the output voltage to 1.0kV...")
hpp120256.voltage_setpoint = 1.0 # kV
# Now HV output should be rising to reach the 1.0kV at 50.0 V/s
hpp120256.wait_for_output_voltage_reached(abs_output_voltage_error, 1.0, 40.0)
# Here voltage HV output should be at 1.0 kV
hpp120256.shutdown()
hpp120256.wait_for_output_voltage_reached(abs_output_voltage_error, 1.0, 60.0)
# Here voltage HV output should be at 0.0 kV
inst.output_enabled = False
# Now the HV voltage source is in safe state
```

**class EurotestHPP120256Status**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntFlag

Auxiliary class create for translating the instrument 16bits\_status\_string into an Enum\_IntFlag that will help to the user to understand such status.

## ask(command)

Overrides Instrument ask method for including query\_delay time on parent call. :param command: Command string to be sent to the instrument. :returns: String returned by the device without read\_termination.

## property current

Measure the actual output current in mAmps (float).

### property current\_limit

Control the current limit in mAmps (float strictly from 0 to 25).

### property current\_range

Measure the actual output current range in mAmps (float).

### emergency\_off()

The output of the HV source will be switched OFF permanently and the values of the voltage and current settings set to zero

#### property id

Get the identification of the instrument (string)

#### property kill\_enabled

Control the instrument kill enable (boolean).

# property lam\_status

Get the instrument lam status (string).

### property output\_enabled

Control the instrument output enable (boolean).

# ramp\_to\_zero(voltage\_rate=200.0)

Sets the voltage output setting to zero and the ramp setting to a value determined by the voltage\_rate parameter. In summary, the method conducts (ramps) the voltage output to zero at a determinated voltage changing rate (ramp in V/s). :param voltage\_rate: Is the changing rate (ramp in V/s) for the ramp setting

### **shutdown**(voltage\_rate=200.0)

Change the output voltage setting (V) to zero and the ramp speed - voltage\_rate (V/s) of the output voltage. After calling shutdown, if the HV voltage output > 0 it should drop to zero at a certain rate given by the voltage\_rate parameter. :param voltage\_rate: indicates the changing rate (V/s) of the voltage output

#### property status

Get the instrument status (EurotestHPP120256Status).

#### property voltage

Measure the actual output voltage in kVolts (float).

# property voltage\_ramp

Control the voltage ramp in Volts/second (int strictly from 10 to 3000).

#### property voltage\_range

Measure the actual output voltage range in kVolts (float).

### property voltage\_setpoint

Control the voltage set-point in kVolts (float strictly from 0 to 12).

7.23. EURO TEST 255

Wait until HV voltage output reaches the voltage setpoint.

Checks the voltage output every check\_period seconds and raises an exception if the voltage output doesn't reach the voltage setting until the timeout time. :param voltage\_setpoint: the voltage in kVolts setted in the HV power supply which should be present at the output after some time (depends on the ramp setting). :param abs\_output\_voltage\_error: absolute error in kVolts for being considered an output voltage reached. :param check\_period: voltage output will be measured every check\_period (seconds) time. :param timeout: time (seconds) give to the voltage output to reach the voltage setting. :return: None :raises: Exception if the voltage output can't reach the voltage setting before the timeout completes (seconds).

```
write(command, **kwargs)
```

Overrides Instrument write method for including write\_delay time after the parent call.

#### Parameters

**command** – command string to be sent to the instrument

# 7.24 Fluke

This section contains specific documentation on the Fluke instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.24.1 Fluke 7341 Temperature bath

class pymeasure.instruments.fluke.Fluke7341(adapter, name='Fluke7341', \*\*kwargs)

Bases: Instrument

Represents the compact constant temperature bath from Fluke.

### property id

Get the instrument model.

#### read()

Read up to (excluding) *read\_termination* or the whole read buffer.

Extract the value from the response string.

Responses are in the format "type: value optional information". Optional information is for example the unit (degree centigrade or Fahrenheit).

#### property set\_point

Control the temperature setpoint (float from -40 to 150 °C) The unit is as defined in property unit.

### property temperature

Measure the current bath temperature. The unit is as defined in property unit.

### property unit

Control the temperature unit: *c* for Celsius and *f* for Fahrenheit`.

# 7.25 F.W. Bell

This section contains specific documentation on the F.W. Bell instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.25.1 F.W. Bell 5080 Handheld Gaussmeter

Bases: SCPIMixin, Instrument

Represents the F.W. Bell 5080 Handheld Gaussmeter and provides a high-level interface for interacting with the instrument

#### **Parameters**

port - The serial port of the instrument

```
meter = FWBell5080('/dev/ttyUSB0') # Connects over serial port /dev/ttyUSB0 (Linux)

meter.units = 'gauss' # Sets the measurement units to Gauss
meter.range = 1 # Sets the range to 3 kG
print(meter.field) # Reads and prints a field measurement in G

fields = meter.fields(100) # Samples 100 field measurements
print(fields.mean(), fields.std()) # Prints the mean and standard deviation of the samples
```

### auto\_range()

Enables the auto range functionality.

### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

## check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

7.25. F.W. Bell 257

#### clear()

Clear the instrument status byte.

### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property field

Measure the field in the appropriate units (float).

# fields(samples=1)

Returns a numpy array of field samples for a given sample number.

#### Parameters

**samples** – The number of samples to preform

#### property id

Get the identification of the instrument.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

### property options

Get the device options installed.

## property range

Control the maximum field range in the active units (int). The range unit is dependent on the current units mode (gauss, tesla, amp-meter). Value sets an equivalent range across units that increases in magnitude (1, 10, 100).

Value	gauss	tesla	amp-meter
0	300 G	30 mT	23.88 kAm
1	3 kG	300 mT	238.8 kAm
2	30 kG	3 T	2388 kAm

# read()

Overwrites the *Instrument.read* method to remove semicolons and replace spaces with colons.

#### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

## read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Resets the instrument.

## shutdown()

Brings the instrument to a safe and stable state

#### property status

Get the status byte and Master Summary Status bit.

### property units

Get the field units (str), which can take the values: 'gauss', 'gauss ac', 'tesla', 'tesla ac', 'amp-meter', and 'amp-meter ac'. The AC versions configure the instrument to measure AC.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- command command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

### **Parameters**

- command Command to send.
- values The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.26 Heidenhain

This section contains specific documentation on the Heidenhain instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.26.1 Heidenhain ND287 Position Display Unit

Bases: Instrument

Represents the Heidenhain ND287 position display unit used to readout and display absolute position measured by Heidenhain encoders.

7.26. Heidenhain 259

#### check\_errors()

Method to read an error status message and log when an error is detected.

#### Returns

String with the error message as its contents.

# property id

Get the string identification property for the device.

## property position

Measure the encoder's current position (float). Note that the get\_process performs a mapping from the returned value to a float measured in the units specified by *ND287.units*. The get\_process is modified dynamically as this mapping changes slightly between different units.(dynamic)

### property status

Get the encoder's status bar

# property units

Control the unit of measure set on the device. Valid values are 'mm' and 'inch' Note that this parameter can only be set manually on the device. So this argument only ensures that the instance units and physical device settings match. I.e., this property does not change any physical device setting.

# 7.27 HC Photonics

This section contains specific documentation on the HC Photonics instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.27.1 HCP TC038 crystal oven

class pymeasure.instruments.hcp.TC038(adapter, name='TC038', address=1, timeout=1000, \*\*kwargs)

Bases: Instrument

Communication with the HCP TC038 oven.

This is the older version with an AC power supply and AC heater.

It has parity or framing errors from time to time. Handle them in your application.

The oven always responds with an "OK" to all valid requests or commands.

### **Parameters**

- **adapter** (*str*) Name of the COM-Port.
- address (int) Address of the device. Should be between 1 and 99.
- **timeout** (*int*) Timeout in ms.

## check\_set\_errors()

Check for errors after having set a property.

Called if check\_set\_errors=True is set for that property.

### property information

Get the information about the device and its capabilities.

#### property monitored\_value

Measure the currently monitored value. For default it is the current temperature in °C.

## read()

Do error checking on reading.

## set\_monitored\_quantity(quantity='temperature')

Configure the oven to monitor a certain quantity.

quantity may be any key of registers. Default is the current temperature in °C.

#### property setpoint

Control the setpoint of the temperature controller in °C.

### property temperature

Measure the current temperature in °C.

#### write(command)

Send a *command* in its own protocol.

# 7.27.2 HCP TC038D crystal oven

class pymeasure.instruments.hcp.TC038D(adapter, name='TC038D', address=1, timeout=1000, \*\*kwargs)

Bases: Instrument

Communication with the HCP TC038D oven.

This is the newer version with DC heating.

The oven expects raw bytes written, no ascii code, and sends raw bytes. For the variables are two or four-byte modes available. We use the four-byte mode addresses. In that case element count has to be double the variables read.

## check\_set\_errors()

Check for errors after having set a property.

Called if check\_set\_errors=True is set for that property.

### ping(test\_data=0)

Test the connection sending an integer up to 65535, checks the response.

#### read()

Read response and interpret the number, returning it as a string.

### property setpoint

Control the setpoint of the oven in °C.

### property temperature

Measure the current oven temperature in °C.

#### write(command)

Write a command to the device.

# **Parameters**

**command** (str) – comma separated string of: - the function: read ('R') or write ('W') or 'echo', - the address to write to (e.g. '0x106' or '262'), - the values (comma separated) to write - or the number of elements to read (defaults to 1).

7.27. HC Photonics 261

# 7.28 Hewlett Packard

This section contains specific documentation on the Hewlett Packard instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

If the instrument you are looking for is not here, also check Agilent or Keysight for newer instruments.

# 7.28.1 HP 33120A Arbitrary Waveform Generator

Bases: SCPIUnknownMixin, Instrument

Represents the Hewlett Packard 33120A Arbitrary Waveform Generator and provides a high-level interface for interacting with the instrument.

# property amplitude

Control the voltage amplitude of the output signal. The default units are in peak-to-peak Volts, but can be controlled by <code>amplitude\_units</code>. The allowed range depends on the waveform shape and can be queried with <code>max\_amplitude</code> and <code>min\_amplitude</code>. (float)

### property amplitude\_units

Control the units of the amplitude, which can take the values Vpp, Vrms, dBm, and default. (str)

### beep()

Causes a system beep.

### property burst\_count

Control the number of cycles per burst (1 to 50,000 cycles)

### property burst\_enabled

Control state of burst modulation

### property burst\_phase

Control the starting phase angle of a burst (-360 to +360 degrees)

# property burst\_rate

Control the burst rate in Hz fo an internal burst source

### property burst\_source

Control internal or external gate source for burst modulation

### property frequency

Control the frequency of the output in Hz. The allowed range depends on the waveform shape and can be queried with *max\_frequency* and *min\_frequency*. (float)

#### property max\_amplitude

Get the maximum amplitude in Volts for the given shape

#### property max\_burst\_count

Get the maximum burst\_count

# property max\_burst\_phase

Get the maximum burst\_phase

#### property max\_burst\_rate

Get the maximum burst\_rate

# property max\_frequency

Get the maximum frequency in Hz for the given shape

### property max\_offset

Get the maximum offset in Volts for the given shape

#### property min\_amplitude

Get the minimum amplitude in Volts for the given shape

#### property min\_burst\_count

Get the minimum burst\_count

## property min\_burst\_phase

Get the minimum burst\_phase

### property min\_burst\_rate

Get the minimum burst\_rate

#### property min\_frequency

Get the minimum frequency in Hz for the given shape

## property min\_offset

Get the minimum *offset* in Volts for the given shape

### property offset

Control the amplitude voltage offset in Volts. The allowed range depends on the waveform shape and can be queried with <code>max\_offset</code> and <code>min\_offset</code>.

# property shape

Control the shape of the wave, which can take the values: sinusoid, square, triangle, ramp, noise, dc, and user. (str)

# 7.28.2 HP 34401A Multimeter

class pymeasure.instruments.hp.HP34401A(adapter, name='HP 34401A', \*\*kwargs)

Bases: SCPIUnknownMixin, Instrument

Represents the HP / Agilent / Keysight 34401A Multimeter and provides a high-level interface for interacting with the instrument.

```
dmm = HP34401A("GPIB::1")
dmm.function_ = "DCV"
print(dmm.reading) # -> Single float reading

dmm.nplc = 0.02
dmm.autozero_enabled = False
dmm.trigger_count = 100
dmm.trigger_delay = "MIN"
print(dmm.reading) # -> Array of 100 very fast readings
```

### property auto\_input\_impedance\_enabled

Control if automatic input resistance mode is enabled.

Only valid for dc voltage measurements. When disabled (default), the input resistance is fixed at 10 MOhms for all ranges. With AUTO ON, the input resistance is set to >10 GOhms for the 100 mV, 1 V, and 10 V ranges.

# property autorange

Control the autorange state for the currently active function.

### property autozero\_enabled

Control the autozero state.

# beep()

This command causes the multimeter to beep once.

# property beeper\_enabled

Control whether the beeper is enabled.

## property current\_ac

AC current, in Amps

Deprecated since version 0.12: Use the function\_ and reading properties instead.

# property current\_dc

DC current, in Amps

Deprecated since version 0.12: Use the function\_ and reading properties instead.

### property detector\_bandwidth

Control the lowest frequency expected in the input signal in Hertz.

Valid values: 3, 20, 200, "MIN", "MAX".

## property display\_enabled

Control the display state.

#### property displayed\_text

Control the text displayed on the multimeter's display.

The text can be up to 12 characters long; any additional characters are truncated my the multimeter.

### property function\_

Control the measurement function.

Allowed values: "DCV", "DCV\_RATIO", "ACV", "DCI", "ACI", "R2W", "R4W", "FREQ", "PERIOD", "CONTINUITY", "DIODE".

### property gate\_time

Control the gate time (or aperture time) for frequency or period measurements.

Valid values: 0.01, 0.1, 1, "MIN", "MAX". Specifically: 10 ms (4.5 digits), 100 ms (default; 5.5 digits), or 1 second (6.5 digits).

#### init\_trigger()

Set the state of the triggering system to "wait-for-trigger".

Measurements will begin when the specified trigger conditions are satisfied after this command is received.

#### property nplc

Control the integration time in number of power line cycles (NPLC).

Valid values: 0.02, 0.2, 1, 10, 100, "MIN", "MAX". This command is valid only for dc volts, ratio, dc current, 2-wire ohms, and 4-wire ohms.

#### property range\_

Control the range for the currently active function.

For frequency and period measurements, ranging applies to the signal's input voltage, not its frequency

### property reading

Take a measurement of the currently selected function.

Reading this property is equivalent to calling *init\_trigger()*, waiting for completion and fetching the reading(s).

### property remote\_control\_enabled

Control whether remote control is enabled.

### property remote\_lock\_enabled

Control whether the beeper is enabled.

### property resistance

Resistance, in Ohms

Deprecated since version 0.12: Use the function\_ and reading properties instead.

### property resistance\_4w

Four-wires (remote sensing) resistance, in Ohms

Deprecated since version 0.12: Use the function\_ and reading properties instead.

### property resolution

Control the resolution of the measurements.

Not valid for frequency, period, or ratio. Specify the resolution in the same units as the measurement function, not in number of digits. Results in a "Settings Conflict" error if autorange is enabled. MIN selects the smallest value accepted, which gives the most resolution. MAX selects the largest value accepted which gives the least resolution.

## property sample\_count

Controls the number of samples per trigger event.

Valid values: 1 to 50000, "MIN", "MAX".

# property scpi\_version

The SCPI version of the multimeter.

### property self\_test\_result

Initiate a self-test of the multimeter and return the result.

Be sure to set an appropriate connection timeout, otherwise the command will fail.

### property stored\_reading

Measure the reading(s) currently stored in the multimeter's internal memory.

Reading this property will NOT initialize a trigger. If you need that, use the *reading* property instead.

### property stored\_readings\_count

The number of readings currently stored in the internal memory.

#### property terminals\_used

Query the multimeter to determine if the front or rear input terminals are selected.

Returns "FRONT" or "REAR".

# property trigger\_auto\_delay\_enabled

Control the automatic trigger delay state.

If enabled, the delay is determined by function, range, integration time, and ac filter setting. Selecting a specific trigger delay value automatically turns off the automatic trigger delay.

### property trigger\_count

Control the number of triggers accepted before returning to the "idle" state.

Valid values: 1 to 50000, "MIN", "MAX", "INF". The INFinite parameter instructs the multimeter to continuously accept triggers (you must send a device clear to return to the "idle" state).

### property trigger\_delay

Control the trigger delay in seconds.

Valid values (incl. floats): 0 to 3600 seconds, "MIN", "MAX".

# trigger\_single\_autozero()

Trigger an autozero measurement.

Consequent autozero measurements are disabled.

# property trigger\_source

Control the trigger source.

Valid values: "IMM", "BUS", "EXT" The multimeter will accept a software (bus) trigger, an immediate internal trigger (this is the default source), or a hardware trigger from the rear-panel Ext Trig (external trigger) terminal.

# property voltage\_ac

AC voltage, in Volts

Deprecated since version 0.12: Use the function\_ and reading properties instead.

#### write(command)

Write a command to the instrument.

# 7.28.3 HP 3437A System-Voltmeter

class pymeasure.instruments.hp.HP3437A(adapter, name='Hewlett-Packard HP3437A', \*\*kwargs)

Bases: HPLegacyInstrument

Represents the Hewlett Packard 3737A system voltmeter and provides a high-level interface for interacting with the instrument.

**class** SRQ(value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntFlag

Enum element for SRQ mask bit decoding

#### property SRQ\_mask

Return current SRQ mask, this property can be set,

bit assignment for SRQ:

Bit (dec)	Description
1	SRQ when invalid program
2	SRQ when trigger is ignored
4	SRQ when data ready

#### check\_errors()

As this instrument does not have a error indication bit, this function always returns an empty list.

## property delay

Return the value (float) for the delay between two measurements, this property can be set,

valid range: 100ns - 0.999999s

# property number\_readings

Return value (int) for the number of consecutive measurements, this property can be set, valid range: 0 -9999

## pb\_desc

alias of PackedBits

### property range

Return the current measurement voltage range.

This property can be set, valid values: 0.1, 1, 10 (V).



## 1 Note

This instrument does not have autorange capability.

Overrange will be in indicated as 0.99,9.99 or 99.9

## read\_data()

Reads measured data from instrument, returns a np.array.

(This function also takes care of unpacking the data if required)

## Return data

np.array containing the data

#### status\_desc

alias of Status

# property talk\_ascii

A boolean property, True if the instrument is set to ASCII-based communication. This property can be set.

# property trigger

Return current selected trigger mode, this property can be set,

Possible values are:

Value	Explanation
internal	automatic trigger (internal)
external	external trigger (connector on back or GET)
hold/manual	holds the measurement/issues a manual trigger

# 7.28.4 HP 3478A Multimeter

class pymeasure.instruments.hp.HP3478A(adapter, name='Hewlett-Packard HP3478A', \*\*kwargs)

Bases: HPLegacyInstrument

Represents the Hewlett Packard 3478A 5 1/2 digit multimeter and provides a high-level interface for interacting with the instrument.

**class ERRORS**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntFlag

Enum element for errror bit decoding

## property SRQ\_mask

Return current SRQ mask, this property can be set,

bit assigment for SRQ:

Bit (dec)	Description
1	SRQ when Data ready
4	SRQ when Syntax error
8	SRQ when internal error
16	front panel SQR button
32	SRQ by invalid calibration

## property active\_connectors

Return selected connectors ("front"/"back"), based on front-panel selector switch

#### property auto\_range\_enabled

Property describing the auto-ranging status

Value	Status
True	auto-range function activated
False	manual range selection / auto-range disabled

The range can be set with the range property

# property auto\_zero\_enabled

Return auto-zero status, this property can be set

Value	Status
True	auto-zero active
False	auto-zero disabled

### property calibration\_data

Read or write the calibration data as an array of 256 values between 0 and 15.

The calibration data of an HP 3478A is stored in a 256x4 SRAM that is permanently powered by a 3v Lithium battery. When the battery runs out, the calibration data is lost, and recalibration is required.

When read, this property fetches and returns the calibration data so that it can be backed up.

When assigned a value, it similarly expects an array of 256 values between 0 and 15, and writes the values back to the instrument.

When writing, exceptions are raised for the following conditions:

- The CAL ENABLE switch at the front of the instrument is not set to ON.
- The array with values does not contain exactly 256 elements.
- The array with values does not pass a verification check.

IMPORTANT: changing the calibration data results in permanent loss of the previous data. Use with care!

# property calibration\_enabled

Return calibration enable switch setting, based on front-panel selector switch

Value	Status
True	calbration possible
False	calibration locked

#### check\_errors()

Method to read the error status register

### Return error\_status

one byte with the error status register content

#### Rtype error\_status

int

# display\_reset()

Reset the display of the instrument.

#### property display\_text

Displays up to 12 upper-case ASCII characters on the display.

### property display\_text\_no\_symbol

Displays up to 12 upper-case ASCII characters on the display and disables all symbols on the display.

# property error\_status

Checks the error status register

#### property measure\_ACI

Returns the measured value for AC current as a float in A.

### property measure\_ACV

Returns the measured value for AC Voltage as a float in V.

# property measure\_DCI

Returns the measured value for DC current as a float in A.

#### property measure\_DCV

Returns the measured value for DC Voltage as a float in V.

## property measure\_R2W

Returns the measured value for 2-wire resistance as a float in Ohm.

### property measure\_R4W

Returns the measured value for 4-wire resistance as a float in Ohm.

## property measure\_Rext

Returns the measured value for extended resistance mode (>30M, 2-wire) resistance as a float in Ohm.

## property mode

Return current selected measurement mode, this propery can be set. Allowed values are

Mode	Function
ACI	AC current
ACV	AC voltage
DCI	DC current
DCV	DC voltage
R2W	2-wire resistance
R4W	4-wire resistance
Rext	extended resistance method (requires additional 10 M resistor)

## property range

Returns the current measurement range, this property can be set.

Valid values are:

Mode	Range
ACI	0.3, 3, auto
ACV	0.3, 3, 30, 300, auto
DCI	0.3, 3, auto
DCV	0.03, 0.3, 3, 30, 300, auto
R2W	30, 300, 3000, 3E4, 3E5, 3E6, 3E7, auto
R4W	30, 300, 3000, 3E4, 3E5, 3E6, 3E7, auto
Rext	3E7, auto

# property resolution

Returns current selected resolution, this property can be set.

Possible values are 3,4 or 5 (for 3 1/2, 4 1/2 or 5 1/2 digits of resolution)

# status\_desc

alias of Status

# property trigger

Return current selected trigger mode, this property can be set

Possibe values are:

Value	Meaning
auto	automatic trigger (internal)
internal	automatic trigger (internal)
external	external trigger (connector on back or GET)
hold	holds the measurement
fast	fast trigger for AC measurements

# verify\_calibration\_data(cal\_data)

Verify the checksums of all calibration entries.

Expects an array of 256 values with calibration data.

## Return calibration\_correct

True when all checksums are correct.

### Rtype calibration\_correct

boolean

# verify\_calibration\_entry(cal\_data, entry\_nr)

Verify the checksum of one calibration entry.

Expects an array of 256 values with calibration data, and an entry number from 0 to 18.

Returns True when the checksum of the specified calibration entry is correct.

# write\_calibration\_data(cal\_data, verify\_calibration\_data=True)

Method to write calibration data.

The cal\_data parameter format is the same as the calibration\_data property.

Verification of the cal\_data array can be bypassed by setting verify\_calibration\_data to False.

# 7.28.5 HP 8116A 50 MHz Pulse/Function Generator

class pymeasure.instruments.hp.HP8116A(adapter, name='Hewlett-Packard 8116A', \*\*kwargs)

Bases: Instrument

Represents the Hewlett-Packard 8116A 50 MHz Pulse/Function Generator and provides a high-level interface for interacting with the instrument. The resolution for all floating point instrument parameters is 3 digits.

Bases: Enum

Enum of the digits used with the autovernier (see HP8116A.start\_autovernier()).

**class Direction**(*value*, *names=<not given>*, \**values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

Enum of the directions used with the autovernier (see HP8116A.start\_autovernier()).

## GPIB\_trigger()

Initiate trigger via low-level GPIB-command (aka GET - group execute trigger).

# property amplitude

Control the amplitude of the output in V (strict float from 10e-3 to 16). The allowed amplitude range is also limited by the current offset.

#### **ask**(*command*, *num\_bytes=None*)

Write a command to the instrument, read the response, and return the response as ASCII text.

#### **Parameters**

- **command** The command to send to the instrument.
- **num\_bytes** The number of bytes to read from the instrument. If not specified, the number of bytes is automatically determined by the command.

## property autovernier\_enabled

Control whether the autovernier is enabled (bool).

### property burst\_number

Control the number of periods generated in a burst (strict int from 1 to 1999). It is only valid for units with Option 001 in one of the burst modes.

### check\_errors()

Check for errors in the 8116A.

#### Returns

list of error entries or empty list if no error occurred.

# property complement\_enabled

Control whether the complement of the signal is generated (bool).

### property complete

Get whether the measurement is complete (bool).

# property control\_mode

Control the control mode of the instrument. Possible values are 'off', 'FM', 'AM', 'PWM', 'VCO'.

#### property duty\_cycle

Control the duty cycle of the output in percent (float). The allowed range generally is 10 % to 90 %, but it also depends on the current frequency. It is valid for all shapes except 'pulse', where <code>pulse\_width</code> is used instead.

### property frequency

Control the frequency of the output in Hz (strict float from 1e-3 to 52.5e6).

#### property haversine\_enabled

Control whether a haversine/havertriangle signal is generated when in 'triggered', 'internal\_burst' or 'external\_burst' operating mode.

# property high\_level

Control the high level of the output in V (strict float from -7.9 to 8). The allowed high level range must be at least 10 mV greater than the low level.

#### property limit\_enabled

Control whether parameter limiting is enabled (bool).

### property low\_level

Control the low level of the output in V (strict float from -8 to 7.9). The allowed low level range must be at least 10 mV less than the high level.

# property offset

Control the offset of the output in V (strit float from -7.95 to 7.95). The allowed offset range is also limited by the amplitude.

### property operating\_mode

Control the operating mode of the instrument. Possible values (without Option 001) are: 'normal', 'triggered', 'gate', 'external\_width'. With Option 001, 'internal\_sweep', 'external\_sweep', 'external\_width', 'external pulse' are also available.

# property options

Get the device options installed. The only possible option is 001.

### property output\_enabled

Control whether the output is enabled (bool).

## property pulse\_width

Control the pulse width in s (strict float from 8e-9 to 999e-3). The pulse width may not be larger than the period.

## property repetition\_rate

Control the repetition rate in s (i.e. the time between bursts) in 'internal\_burst' mode (strict float from 20e-9 to 999e-3).

#### reset()

Initatiate a reset (like a power-on reset) of the 8116A.

## property shape

Control the shape of the output waveform. Possible values are: 'dc', 'sine', 'triangle', 'square', 'pulse'.

# shutdown()

Gracefully close the connection to the 8116A.

# start\_autovernier(control, digit, direction, start\_value=None)

Start the autovernier on the specified control.

### **Parameters**

- **control** The control to change, pass as HP8116A. some\_control. Allowed controls are frequency, amplitude, offset, duty\_cycle, and pulse\_width
- **digit** The digit to change, type: *HP8116A.Digit*.
- direction The direction in which to change the control, type: HP8116A.Direction.
- start\_value An optional value to start the autovernier at. If not specified, the current
  value of the control is used.

#### property status

Get the status byte of the 8116A as a Status IntFlag-type enum.

# property sweep\_marker\_frequency

Control the frequency marker in both sweep modes in Hz (strict float from 1e-3 to 52.5e6). At this frequency, the marker output switches from low to high.

### property sweep\_start

Control the start frequency in both sweep modes in Hz (strict float from 1e-3 to 52.5e6).

# property sweep\_stop

Control the stop frequency in both sweep modes in Hz (strict float from 1e-3 to 52.5e6).

#### property sweep\_time

Control the sweep time per decade in both sweep modes in s (float). The sweep time is selectable in a 1-2-5 sequence between 10 ms and 500 s.

## property trigger\_slope

Control the slope the trigger triggers on. Possible values are: 'off', 'positive', 'negative'.

#### write(command)

Write a command to the instrument and wait until the 8116A has interpreted it.

# 7.28.6 HP 8560A / 8561B Spectrum Analyzer

Every unit is used in the base unit, so for time it is s (Seconds), frequency in Hz (Hertz) etc...

# **Generic Specific Attributes & Methods**

### **Content**

- General
- Demodulation
- Frequency
- Resolution Bandwidth
- Video
- FFT & Measurements
- Trace
- Marker
- Diagnostic Values
- Sweep
- Normalization
- Open/Short Calibration (Reflection)
- Thru Calibration

# General

## HP856Xx.preset()

Set the spectrum analyzer to a known, predefined state.

'preset' does not affect the contents of any data or trace registers or stored preselector data. 'preset' does not clear the input or output data buffers;

#### HP856Xx.attenuation

Control the input attenuation in decade steps from 10 to 70 db (type 'int') or set to 'AUTO' and 'MAN'(ual)

Type: str, int

```
instr.attenuation = 'AUTO'
instr.attenuation = 60
```

### HP856Xx.amplitude\_unit

Control the amplitude unit with a selection of the following parameters: string 'DBM', 'DBMV', 'DBUV', 'V', 'W', 'AUTO', 'MAN' or use the enum *AmplitudeUnits* 

Type: str

```
instr.amplitude_unit = 'dBmV'
instr.amplitude_unit = AmplitudeUnits.dBmV
```

## HP856Xx.trigger\_mode

Control the trigger mode. Selected trigger conditions must be met in order for a sweep to occur. For the available modes refer to *TriggerMode*. When any trigger mode other than free run is selected, a "T" appears on the left edge of the display.

## HP856Xx.detector\_mode

Control the IF detector used for acquiring measurement data. This is normally a coupled function, in which the spectrum analyzer selects the appropriate detector mode. Four modes are available: normal, positive, negative, and sample.

Type: str

Takes a representation of the detector mode, either from DetectionModes or use 'NEG', 'NRM', 'POS', 'SMP'

```
instr.detector_mode = DetectionModes.SMP
instr.detector_mode = 'NEG'

if instr.detector_mode == DetectionModes.SMP:
    pass
```

#### HP856Xx.coupling

Control the input coupling of the spectrum analyzer. AC coupling protects the input of the analyzer from damaging dc signals, while limiting the lower frequency-range to 100 kHz (although the analyzer will tune down to 0 Hz with signal attenuation).

Type: str

Takes a representation of the coupling mode, either from CouplingMode or use 'AC' / 'DC'

```
instr.coupling = 'AC'
instr.coupling = CouplingMode.DC

if instr.coupling == CouplingMode.DC:
    pass
```

# HP856Xx.set\_auto\_couple()

Set the video bandwidth, resolution bandwidth, input attenuation, sweep time, and center frequency step-size to coupled mode.

These functions can be recoupled individually or all at once. The spectrum analyzer chooses appropriate values for these functions. The video bandwidth and resolution bandwidth are set according to the coupled ratios stored under resolution\_bandwidth\_to\_span\_ratio and video\_bandwidth\_to\_resolution\_bandwidth. If no ratios are chosen, default ratios (1.0 and 0.011, respectively) are used instead.

### HP856Xx.set\_linear\_scale()

Set the spectrum analyzers display to linear amplitude scale.

Measurements made on a linear scale can be read out in any units.

### HP856Xx.logarithmic\_scale

Control the logarithmic amplitude scale. When in linear mode, querying 'logarithmic\_scale' returns a "0". Allowed values are 0, 1, 2, 5, 10

Type: int

```
if instr.logarithmic_scale:
    pass

# set the scale to 10 db per division
instr.logarithmic_scale = 10
```

### HP856Xx.threshold

Control the minimum amplitude level and clips data at this value. Default value is -90 dBm. See also - marker\_threshold does not clip data below its threshold

Type: str, float range -200 to 30



When a trace is in max-hold mode, if the threshold is raised above any of the trace data, the data below the threshold will be permanently lost.

#### HP856Xx.set\_title(string)

Sets character data in the title area of the display, which is in the upper-right corner.

A title can be up to two rows of sixteen characters each, Carriage return and line feed characters are not allowed.

#### HP856Xx.status

Get the decimal equivalent of the bits set in the status byte (see the RQS and SRQ commands). STB is equivalent to a serial poll command. The RQS and associated bits are cleared in the same way that a serial poll command would clear them.

# HP856Xx.check\_done()

Return when all commands in a command string entered before :meth:'check\_done' has been completed. Sending a <code>trigger\_sweep()</code> command before 'check\_done' ensures that the spectrum analyzer will complete a full sweep before continuing on in a program. Depending on the timeout a timeout error from the adapter will raise before the spectrum analyzer can finish due to an extreme long sweep time.

```
instr.trigger_sweep()

# wait for a full sweep and than 'do_something'
instr.check_done()
do_something()
```

# HP856Xx.request\_service(input)

Triggers a service request. This command allows you to force a service request and test a program designed to handle service requests. However, the service request can be triggered only if it is first masked using the request\_service\_conditions command.

#### **Parameters**

**input** (*StatusRegister*) – Bits to emulate a service request

#### HP856Xx.errors

Get a list of errors present (of type *ErrorCode*). An empty list means there are no errors. Reading 'errors' clears all HP-IB errors. For best results, enter error data immediately after querying for errors.

Type: ErrorCode

```
errors = instr.errors
if len(errors) > 0:
    print(errors[0].code)

for error in errors:
    print(error)

if ErrorCode(112) in errors:
    print("yeah")
```

Example result of this python snippet:

```
112
ErrorCode("??CMD?? - Unrecognized command")
ErrorCode("NOP NUM - Command cannot have numeric units")
yeah
```

#### HP856Xx.save\_state(inp)

Saves the currently displayed instrument state in the specified state register.

#### **Parameters**

- inp State to be recalled: either storage slot 0 ... 9 or 'LAST' or 'PWRON'
- **inp** str, int

```
instr.preset()
instr.center_frequency = 300e6
instr.span = 20e6
instr.save_state("PWRON")
```

## HP856Xx.recall\_state(inp)

Set to the display a previously saved instrument state. See save\_state().

# **Parameters**

- inp State to be recalled: either storage slot 0 ... 9 or 'LAST' or 'PWRON'
- **inp** str, int

```
instr.save_state(7)
instr.preset()
instr.recall_state(7)
```

# HP856Xx.request\_service\_conditions

Control a bit mask that specifies which service requests can interrupt a program sequence.

```
instr.request_service_conditions = StatusRegister.ERROR_PRESENT | StatusRegister.

→TRIGGER

print(instr.request_service_conditions)
StatusRegister.ERROR_PRESENT|TRIGGER
```

```
HP856Xx.set_maximum_hold = <function HP856Xx.set_maximum_hold>
```

HP856Xx.set\_minimum\_hold = <function HP856Xx.set\_minimum\_hold>

## HP856Xx.reference\_level\_calibration

Control the calibration of the reference level remotely and retuns the current calibration. To calibrate the reference level, connect the 300 MHz calibration signal to the RF input. Set the center frequency to 300 MHz, the frequency span to 20 MHz, and the reference level to -10 dBm. Use the RLCAL command to move the input signal to the reference level. When the signal peak falls directly on the reference-level line, the reference level is calibrated. Storing this value in the analyzer in EEROM can be done only from the front panel. The RLCAL command, when queried, returns the current value.

# Type: float

```
# connect cal signal to rf input
instr.preset()
instr.amplitude_unit = AmplitudeUnits.DBM
instr.center_frequency = 300e6
instr.span = 100e3
instr.reference_level = 0
instr.trigger_sweep()

instr.peak_search(PeakSearchMode.High)
level = instr.marker_amplitude
rlcal = instr.reference_level_calibration - int((level + 10) / 0.17)
instr.reference_level_calibration = rlcal
```

### HP856Xx.reference\_offset

Control an offset applied to all amplitude readouts (for example, the reference level and marker amplitude). The offset is in dB, regardless of the selected scale and units. The offset can be useful to account for gains of losses in accessories connected to the input of the analyzer. When this function is active, an "R" appears on the left edge of the display.

Type: int

# HP856Xx.reference\_level

Control the reference level, or range level when in normalized mode. (Range level functions the same as reference level.) The reference level is the top horizontal line on the graticule. For best measurement accuracy, place the peak of a signal of interest on the reference-level line. The spectrum analyzer input attenuator is coupled to the reference level and automatically adjusts to avoid compression of the input signal. Refer also to <code>amplitude\_unit</code>. Minimum reference level is -120.0 dBm or 2.2 uV

Type: float

# HP856Xx.display\_line

Control the horizontal display line for use as a visual aid or for computational purposes. The default value is 0 dBm.

Type: float

Takes a value with the unit of amplitude\_unit

```
instr.display_line = -10
if instr.display_line == 0:
    pass
```

#### HP856Xx.protect\_state\_enabled

Control the storing of any new data in the state or trace registers. If set to 'True', the registers are "locked"; the data in them cannot be erased or overwritten, although the data can be recalled. To "unlock" the registers, and store new data, set 'protect\_state\_enabled' to off by selecting 'False' as the parameter.

Type: bool

## HP856Xx.mixer\_level

Control the maximum signal level that is at the input mixer. The attenuator automatically adjusts to ensure that this level is not exceeded for signals less than the reference level. From -80 to -10 DB.

Type: int

## HP856Xx.frequency\_counter\_mode\_enabled

Set the device into a frequency counter mode that counts the frequency of the active marker or the difference in frequency between two markers. If no marker is active, 'frequency\_counter\_mode\_enabled' places a marker at the center of the trace and counts that marker frequency. The frequency counter provides a more accurate frequency reading; it pauses at the marker, counts the value, then continues the sweep. To adjust the frequency counter resolution, use the 'frequency\_counter\_resolution' command. To return the counter value, use the 'marker\_frequency' command.

```
instr.frequency_counter_mode_enabled = True
```

### HP856Xx.frequency\_counter\_resolution

Control the resolution of the frequency counter. Refer to the 'frequency\_counter\_mode' command. The default value is 10 kHz.

Type int

```
# activate frequency counter mode
instr.frequency_counter_mode = True

# adjust resolution to 1 Hz
instr.frequency_counter_resolution = 1

if instr.frequency_counter_resolution:
    pass
```

# HP856Xx.adjust\_all()

Activate the local oscillator (LO) and intermediate frequency (IF) alignment routines. These are the same routines that occur when is switched on. Commands following 'adjust\_all' are not executed until after the analyzer has finished the alignment routines.

# HP856Xx.adjust\_if

Control the automatic IF adjustment. This function is normally on. Because the IF is continuously adjusting, executing the IF alignment routine is seldom necessary. When the IF adjustment is not active, an "A" appears on the left side of the display.

- "FULL" IF adjustment is done for all IF settings.
- "CURR" IF adjustment is done only for the IF settings currently displayed.

- False turns the continuous IF adjustment off.
- True reactivates the continuous IF adjustment.

Type: bool, str

## HP856Xx.hold()

Freeze the active function at its current value.

If no function is active, no operation takes place.

### HP856Xx.annotation\_enabled

Set the display annotation off or on.

Type: bool

### HP856Xx.set\_crt\_adjustment\_pattern()

Activate a CRT adjustment pattern, shown in Figure 5-3. Use the X POSN, Y POSN, and TRACE ALIGN adjustments (available from the rear panel) to align the display. Use X POSN and Y POSN to move the display horizontally and vertically, respectively. Use TRACE ALIGN to straighten a tilted display. To remove the pattern from the screen, execute the *preset()* command.

# HP856Xx.display\_parameters

Get the location of the lower left (P1) and upper right (P2) vertices as a tuple of the display window.

Type: tuple

```
repr(instr.display_parameters)
(72, 16, 712, 766)
```

#### HP856Xx.firmware\_revision

Get the revision date code of the spectrum analyzer firmware.

Type: datetime.date

# HP856Xx.graticule\_enabled

Control the display graticule. Switch it either on or off.

Type: bool

```
instr.graticule = True
if instr.graticule:
    pass
```

## HP856Xx.serial\_number

Get the spectrum analyzer serial number.

#### HP856Xx.id

Get the identification of the device with software and hardware revision (e.g. HP8560A,002, H03)

Type: str

```
print(instr.id)
HP8560A,002,H02
```

# HP856Xx.elapsed\_time

Get the elapsed time (in hours) of analyzer operation. This value can be reset only by Hewlett-Packard.

Type: int

```
print(elapsed_time)
1998
```

#### **Demodulation**

#### HP856Xx.demodulation mode

Control the demodulation mode of the spectrum analyzer. Either AM or FM demodulation, or turns the demodulation — off. Place a marker on a desired signal and then set <code>demodulation\_mode</code>; demodulation takes place on this signal. If no marker is on, <code>demodulation\_mode</code> automatically places a marker at the center of the trace and demodulates the frequency at that marker position. Use the volume and squelch controls to adjust the speaker and listen.

Type: str

Takes a representation of the demodulation mode, either from <code>DemodulationMode</code> or use 'OFF', 'AM', 'FM'

```
instr.demodulation_mode = 'AC'
instr.demodulation_mode = DemodulationMode.AM

if instr.demodulation_mode == DemodulationMode.FM:
    instr.demodulation_mode = Demodulation.OFF
```

### HP856Xx.demodulation\_agc\_enabled

Control the demodulation automatic gain control (AGC). The AGC keeps the volume of the speaker relatively constant during AM demodulation. AGC is available only during AM demodulation and when the frequency span is greater than 0 Hz.

Type: bool

```
instr.demodulation_agc = True
if instr.demodulation_agc:
   instr.demodulation_agc = False
```

# HP856Xx.demodulation\_time

Control the amount of time that the sweep pauses at the marker to demodulate a signal. The default value is 1 second. When the frequency span equals 0 Hz, demodulation is continuous, except when between sweeps. For truly continuous demodulation, set the frequency span to 0 Hz and the trigger mode to single sweep (see TM). Minimum 100 ms to maximum 60 s

Type: float

```
# set the demodulation time to 1.2 seconds
instr.demodulation_time = 1.2

if instr.demodulation_time == 10:
    pass
```

## HP856Xx.squelch

Control the squelch level for demodulation. When this function is on, a dashed line indicating the squelch level appears on the display. A marker must be active and above the squelch line for demodulation to occur. Refer to the <code>demodulation\_mode</code> command. The default value is -120 dBm.

Type: str,int

```
instr.preset()
instr.start_frequency = 88e6
instr.stop_frequency = 108e6

instr.peak_search(PeakSearchMode.High)
instr.demodulation_time = 10

instr.squelch = -60
instr.demodulation_mode = DemodulationMode.FM
```

# **Frequency**

# HP856Xx.start\_frequency

Control the start frequency and set the spectrum analyzer to start-frequency/ stop-frequency mode. If the start frequency exceeds the stop frequency, the stop frequency increases to equal the start frequency plus 100 Hz. The center frequency and span change with changes in the start frequency.

Type: float

```
instr.start_frequency = 300.5e6
if instr.start_frequency == 200e3:
    print("Correct frequency")
```

(dynamic)

# HP856Xx.stop\_frequency

Control the stop frequency and set the spectrum analyzer to start-frequency/ stop-frequency mode. If the stop frequency is less than the start frequency, the start frequency decreases to equal the stop frequency minus 100 Hz. The center frequency and span change with changes in the stop frequency.

Type: float

```
instr.stop_frequency = 300.5e6
if instr.stop_frequency == 200e3:
    print("Correct frequency")
```

(dynamic)

### HP856Xx.center\_frequency

Control the center frequency in hertz and sets the spectrum analyzer to center frequency / span mode.

The span remains constant; the start and stop frequencies change as the center frequency changes.

Type: float

```
instr.center_frequency = 300.5e6
if instr.center_frequency == 200e3:
    print("Correct frequency")
```

(dynamic)

# HP856Xx.frequency\_offset

Control an offset added to the displayed absolute-frequency values, including marker-frequency values.

It does not affect the frequency range of the sweep, nor does it affect relative frequency readouts. When this function is active, an "F" appears on the left side of the display. Changes all the following frequency measurements.

Type: float

```
instr.frequency_offset = 2e6
if instr.frequency_offset == 2e6:
    print("Correct frequency")
```

(dynamic)

# HP856Xx.frequency\_reference\_source

Control the frequency reference source. Select either the internal frequency reference (INT) or supply your own external reference (EXT). An external reference must be 10 MHz (+100 Hz) at a minimum amplitude of 0 dBm. Connect the external reference to J9 (10 MHz REF IN/OUT) on the rear panel. When the external mode is selected, an "X" appears on the left edge of the display.

Type: str

Takes element of FrequencyReference or use 'INT', 'EXT'

```
instr.frequency_reference_source = 'INT'
instr.frequency_reference_source = FrequencyReference.EXT

if instr.frequency_reference_source == FrequencyReference.INT:
    instr.frequency_reference_source = FrequencyReference.EXT
```

#### HP856Xx.span

Control the frequency span. The center frequency does not change with changes in the frequency span; start and stop frequencies do change. Setting the frequency span to 0 Hz effectively allows an amplitude-versus-time mode in which to view signals. This is especially useful for viewing modulation. Querying SP will leave the analyzer in center frequency /span mode.

# HP856Xx.set\_full\_span()

Set the spectrum analyzer to the full frequency span as defined by the instrument.

The full span is 2.9 GHz for the HP 8560A. For the HP 8561B, the full span is 6.5 GHz.

### HP856Xx.frequency\_display\_enabled

Get the state of all annotations that describes the spectrum analyzer frequency. returns 'False' if no annotations are shown and vice versa 'True'. This includes the start and stop frequencies, the center frequency, the frequency span, marker readouts, the center frequency step-size, and signal identification to center frequency. To retrieve the frequency data, query the spectrum analyzer.

Type: bool

```
if instr.frequency_display:
    print("Frequencies get displayed")
```

#### **Resolution Bandwidth**

#### HP856Xx.resolution\_bandwidth

Control the resolution bandwidth. This is normally a coupled function that is selected according to the ratio selected by the RBR command. If no ratio is selected, a default ratio (0.011) is used. The bandwidth, which ranges from 10 Hz to 2 MHz, may also be selected manually.

Type: str, dec

### HP856Xx.resolution\_bandwidth\_to\_span\_ratio

Control the coupling ratio between the resolution bandwidth and the frequency span. When the frequency span is changed, the resolution bandwidth is changed to satisfy the selected ratio. The ratio ranges from 0.002 to 0.10. The "UP" and "DN" parameters adjust the ratio in a 1, 2, 5 sequence. The default ratio is 0.011.

#### **Video**

### HP856Xx.video\_trigger\_level

Control the video trigger level when the trigger mode is set to VIDEO (refer to the trigger\_mode command). A dashed line appears on the display to indicate the level. The default value is 0 dBm. Range -220 to 30.

Type: float

### HP856Xx.video\_bandwidth\_to\_resolution\_bandwidth

Control the coupling ratio between the video bandwidth and the resolution bandwidth. Thus, when the resolution bandwidth is changed, the video bandwidth changes to satisfy the ratio. The ratio ranges from 0.003 to 3 in a 1, 3, 10 sequence. The default ratio is 1. When a new ratio is selected, the video bandwidth changes to satisfy the new ratio—the resolution bandwidth does not change value.

#### HP856Xx.video\_bandwidth

Control the video bandwidth. This is normally a coupled function that is selected according to the ratio selected by the VBR command. (If no ratio is selected, a default ratio, 1.0, is used instead.) Video bandwidth filters (or smooths) post-detected video information. The bandwidth, which ranges from 1 Hz to 3 MHz, may also be selected manually. If the specified video bandwidth is less than 300 Hz and the resolution bandwidth is greater than or equal to 300 Hz, the IF detector is set to sample mode. Reducing the video bandwidth or increasing the number of video averages will usually smooth the trace by about as much for the same total measurement time. Reducing the video bandwidth to one-third or less of the resolution bandwidth is desirable when the number of video averages is above 25. For the case where the number of video averages is very large, and the video bandwidth is equal to the resolution bandwidth, internal mathematical limitations allow about 0.4 dB overresponse to noise on the logarithmic scale. The overresponse is negligible (less than 0.1 dB) for narrower video bandwidths.

Type: int

#### HP856Xx.video\_average

Control the video averaging function. Video averaging smooths the displayed trace without using a narrow bandwidth. 'video\_average' sets the IF detector to sample mode (see the DET command) and smooths the trace by averaging successive traces with each other. If desired, you can change the detector mode during video averaging. Video averaging is available only for trace A, and trace A must be in clear-write mode for 'video\_average' to operate. After 'video\_average' is executed, the number of sweeps that have been averaged appears at the top of the analyzer screen. Using video averaging allows you to view changes to the entire trace much faster than using narrow video filters. Narrow video filters require long sweep times, which may not be desired. Video averaging, though requiring more sweeps, uses faster sweep times; in some cases, it can produce a smooth trace as fast as a video filter.

Type: str, int

#### **FFT & Measurements**

### HP856Xx.create\_fft\_trace\_window(trace, window\_mode)

Creates a window trace array for the fast Fourier transform (FFT) function.

The trace-window function creates a trace array according to three built-in algorithms: UNIFORM, HANNING, and FLATTOP. When used with the FFT command, the three algorithms give resultant passband shapes that represent a compromise among amplitude uncertainty, sensitivity, and frequency resolution. Refer to the FFT command description for more information.

#### **Parameters**

- **trace** (str) A representation of the trace, either from *Trace* or use 'TRA' for Trace A or 'TRB' for Trace B
- window\_mode (str) A representation of the window mode, either from WindowType or use 'HANNING', 'FLATTOP' or 'UNIFORM'

### HP856Xx.get\_power\_bandwidth(trace, percent)

Measure the combined power of all signal responses contained in a trace array. The command then computes the bandwidth equal to a percentage of the total power. For example, if 100% is specified, the power bandwidth equals the current frequency span. If 50% is specified, trace elements are eliminated from either end of the array, until the combined power of the remaining trace elements equals half of the total power computed. The frequency span of these remaining trace elements is the power bandwidth output to the controller.

#### **Parameters**

- **trace** (str) A representation of the trace, either from *Trace* or use 'TRA' for Trace A or 'TRB' for Trace B
- **percent** (*float*) Percentage of total power 0 ... 100 %

```
# reset spectrum analyzer
instr.preset()

# set to single sweep mode
instr.sweep_single()

instr.center_frequency = 300e6
instr.span = 1e6

instr.maximum_hold()

instr.trigger_sweep()

if instr.done:
    pbw = instr.power_bandwidth(Trace.A, 99.0)
    print("The power bandwidth at 99 percent is %f kHz" % (pbw / 1e3))
```

# HP856Xx.do\_fft(source, destination, window)

Calculate and show a discrete Fourier transform.

The FFT command performs a discrete Fourier transform on the source trace array and stores the logarithms of the magnitudes of the results in the destination array. The maximum length of any of the traces is 601 points. FFT is designed to be used in transforming zero-span amplitude-modulation information into the frequency domain. Performing an FFT on a frequency sweep will not provide time-domain results. The FFT results are displayed on the spectrum analyzer in a logarithmic amplitude scale. For the horizontal dimension, the frequency at the left side of the graph is 0 Hz, and at the right side is Finax- Fmax is equal to 300 divided by sweep time. As an

example, if the sweep time of the analyzer is 60 ms, Fmax equals 5 kHz. The FFT algorithm assumes that the sampled signal is periodic with an integral number of periods within the time-record length (that is, the sweep time of the analyzer). Given this assumption, the transform computed is that of a time waveform of infinite duration, formed of concatenated time records. In actual measurements, the number of periods of the sampled signal within the time record may not be integral. In this case, there is a step discontinuity at the intersections of the concatenated time records in the assumed time waveform of infinite duration. This step discontinuity causes measurement errors, both amplitude uncertainty (where the signal level appears to vary with small changes in frequency) and frequency resolution (due to filter shape factor and sidelobes). Windows are weighting functions that are applied to the input data to force the ends of that data smoothly to zero, thus reducing the step discontinuity and reducing measuremen errors.

#### **Parameters**

- source (str) A representation of the trace, either from Trace or use 'TRA' for Trace A
  or 'TRB' for Trace B
- **destination** (str) A representation of the trace, either from *Trace* or use 'TRA' for Trace A or 'TRB' for Trace B
- window (str) A representation of the trace, either from Trace or use 'TRA' for Trace A
  or 'TRB' for Trace B

#### **Trace**

#### HP856Xx.view\_trace(trace)

Display the current contents of the selected trace, but does not update the contents. View mode may be executed before a sweep is complete when <code>sweep\_single()</code> and <code>trigger\_sweep()</code> are not used.

#### **Parameters**

**trace** (str) – A representation of the trace, either from *Trace* or use 'TRA' for Trace A or 'TRB' for Trace B

### Raises

- **TypeError** Type isn't 'string'
- ValueError Value is 'TRA' nor 'TRB'

# HP856Xx.get\_trace\_data\_a()

Get the data of trace A as a list.

The function returns the 601 data points as a list in the amplitude format. Right now it doesn't support the linear scaling due to the manual just being wrong.

# HP856Xx.get\_trace\_data\_b()

Get the data of trace B as a list.

The function returns the 601 data points as a list in the amplitude format. Right now it doesn't support the linear scaling due to the manual just being wrong.

### HP856Xx.set trace data a

Set the trace data of trace A.



### 🛕 Warning

The string based method this attribute is using takes its time. Something around 5000ms timeout at the adapter seems to work well.

### HP856Xx.set\_trace\_data\_b

Set the trace data of trace B also allows to write the data.



### Warning

The string based method this attribute is using takes its time. Something around 5000ms timeout at the adapter seems to work well.

#### HP856Xx.trace\_data\_format

Control the format used to input and output trace data (see the TRA/TRB command, You must specify the desired format when transferring data from the spectrum analyzer to a computer; this is optional when transferring data to the analyzer.

Type: str or TraceDataFormat



### Warning

Only needed for manual read out of trace data. Don't use this if you don't know what You are doing.

### HP856Xx.save\_trace(trace, number)

Saves the selected trace in the specified trace register.

#### **Parameters**

- trace (str) A representation of the trace, either from Trace or use 'TRA' for Trace A or 'TRB' for Trace B
- number (int) Storage location from 0 ... 7 where to store the trace

```
instr.preset()
instr.center_frequency = 300e6
instr.span = 20e6
instr.save_trace(Trace.A, 7)
instr.preset()
# reload - at 7 stored trace - to Trace B
instr.recall_trace(Trace.B, 7)
```

#### HP856Xx.recall\_trace(trace, number)

Recalls previously saved trace data to the display. See save\_trace(). Either as Trace A or Trace B.

#### **Parameters**

- trace (str) A representation of the trace, either from Trace or use 'TRA' for Trace A or 'TRB' for Trace B
- **number** (int) Storage location from 0 ... 7 where to store the trace

```
instr.preset()
instr.center_frequency = 300e6
instr.span = 20e6
instr.save_trace(Trace.A, 7)
                                                                             (continues on next page)
```

(continued from previous page)

```
instr.preset()
# reload - at 7 stored trace - to Trace B
instr.recall_trace(Trace.B, 7)
```

### HP856Xx.clear\_write\_trace(trace)

Set the chosen trace to clear-write mode. This mode sets each element of the chosen trace to the bottom-screen value; then new data from the detector is put in the trace with each sweep.

```
instr.clear_write_trace('TRA')
instr.clear_write_trace(Trace.A)
```

#### **Parameters**

trace (str) – A representation of the trace, either from trace or use 'TRA' for Trace A or 'TRB' for Trace B

#### Raises

- **TypeError** Type isn't 'string'
- ValueError Value is 'TRA' nor 'TRB'

### HP856Xx.subtract\_display\_line\_from\_trace\_b()

Subtract the display line from trace B and places the result in dBm (when in log mode) in trace B, which is then set to view mode.

In linear mode, the results are in volts.

### HP856Xx.exchange\_traces()

Exchange the contents of trace A with those of trace B.

If the traces are in clear-write or max-hold mode, the mode is changed to view. Otherwise, the traces remain in their initial mode.

#### HP856Xx.blank\_trace(trace)

Blank the chosen trace from the display. The current contents of the trace remain in the trace but are not updated.

```
instr.blank_trace('TRA')
instr.blank_trace(Trace.A)
```

### **Parameters**

trace (str) – A representation of the trace, either from Trace or use 'TRA' for Trace A or 'TRB' for Trace B

#### Raises

- **TypeError** Type isn't 'string'
- ValueError Value is 'TRA' nor 'TRB'

### HP856Xx.trace\_a\_minus\_b\_plus\_dl\_enabled

Control subtraction of trace B from trace A and addition to the display line, and stores the result in dBm (when in log mode) in trace A. When in linear mode, the result is in volts. If trace A is in clear-write or max-hold mode, this function is continuous. When this function is active, an "M" appears on the left side of the display.

Type: bool

# Warning

The displayed amplitude of each trace element falls in one of 600 data points. There are 10 points of overrange, which corresponds to one-sixth of a division Kg of overrange. When adding or subtracting trace data, any results exceeding this limit are clipped at the limit.

### HP856Xx.trace\_a\_minus\_b\_enabled

Control subtraction of the contents of trace B from trace A. It places the result, in dBm (when in log mode), in trace A. When in linear mode, the result is in volts. If trace A is in clear-write or max-hold mode, this function is continuous. When AMB is active, an "M" appears on the left side of the display. trace\_a\_minus\_b\_plus\_dl overrides AMB.

Type: bool



### Warning

The displayed amplitude of each trace element falls in one of 600 data points. There are 10 points of overrange, which corresponds to one-sixth of a division Kg of overrange. When adding or subtracting trace data, any results exceeding this limit are clipped at the limit.

#### Marker

### HP856Xx.search\_peak(mode)

Place a marker on the highest point on a trace, the next-highest point, the next-left peak, or the next-right peak. The default is 'HI' (highest point). The trace peaks must meet the criteria of the marker threshold and peak excursion functions in order for a peak to be found. See also the peak\_threshold and peak\_excursion commands.

#### **Parameters**

```
mode (str) – Takes 'HI', 'NH', 'NR', 'NL' or the enumeration PeakSearchMode
```

```
instr.search_peak('NL')
instr.search_peak(PeakSearchMode.NextHigh)
```

### HP856Xx.marker\_amplitude

Get the amplitude of the active marker. If no marker is active, MKA places a marker at the center of the trace and returns that amplitude value. In the amplitude\_unit() unit.

Type: float

```
level = instr.marker_amplitude
unit = instr.amplitude_unit
print("Level: %f %s" % (level, unit))
```

#### HP856Xx.set\_marker\_to\_center\_frequency()

Set the center frequency to the frequency value of an active marker.

# HP856Xx.marker\_delta

Control a second marker on the trace. The parameter value specifies the distance in frequency or time (when in zero span) between the two markers. If queried - returns the frequency or time of the second marker.

Type: float

```
# place second marker 1 MHz apart from the first marker
instr.marker_delta = 1e6

# print frequency of second marker in case it got moved automatically
print(instr.marker_delta)
```

### HP856Xx.marker\_frequency

Control the frequency of the active marker. Default units are in Hertz.

Type: float

```
# place marker no. 1 at 100 MHz
instr.marker_frequency = 100e6

# print frequency of the marker in case it got moved automatically
print(instr.marker_frequency)
```

(dynamic)

### HP856Xx.set\_marker\_minimum()

Place an active marker on the minimum signal detected on a trace.

#### HP856Xx.marker\_noise\_mode\_enabled

Control the detector mode to sample and compute the average of 32 data points (16 points on one side of the marker, the marker itself, and 15 points on the other side of the marker). This average is corrected for effects of the log or linear amplifier, bandwidth shape factor, IF detector, and resolution bandwidth. If two markers are on (whether in 'marker\_delta' mode or 1/marker delta mode), 'marker\_noise\_mode\_enabled' works on the active marker and not on the anchor marker. This allows you to measure signal-to-noise density directly. To query the value, use the 'marker\_amplitude' command.

Type: bool

```
# activate signal-to-noise density mode
instr.marker_noise_mode_enabled = True

# get noise density by `marker_amplitude`
print("Signal-to-noise density: %d dbm / Hz" % instr.marker_amplitude)
```

### HP856Xx.deactivate\_marker(all\_markers=False)

Turn off the active marker or, if specified, turn off all markers.

#### **Parameters**

 ${f all\_markers}\ (bool)$  – If True the call deactivates all markers, if false only the currently active marker (optional)

```
# place first marker at 300 MHz
instr.marker_frequency = 300e6

# place second marker 2 MHz apart from first
instr.marker_delta = 2e6

# deactivate active marker (delta marker)
instr.deactivate_marker()
```

(continues on next page)

(continued from previous page)

```
# deactivate all markers
instr.deactivate_marker(all_markers=True)
```

#### HP856Xx.marker\_threshold

Control the minimum amplitude level from which a peak on the trace can be detected. The default value is -130 dBm. See also the *peak\_excursion* command. Any portion of a peak that falls below the peak threshold is used to satisfy the peak excursion criteria. For example, a peak that is equal to 3 dB above the threshold when the peak excursion is equal to 6 dB will be found if the peak extends an additional 3 dB or more below the threshold level. Maximum 30 db to minimum -200 db.

Type: signed int

```
instr.marker_threshold = -70
if instr.marker_threshold > -80:
    pass
```

#### HP856Xx.peak\_excursion

Control what constitutes a peak on a trace. The chosen value specifies the amount that a trace must increase monotonically, then decrease monotonically, in order to be a peak. For example, if the peak excursion is 10 dB, the amplitude of the sides of a candidate peak must descend at least 10 dB in order to be considered a peak (see Figure 5-4) The default value is 6 dB. In linear mode, enter the marker peak excursion as a unit-less number. Any portion of a peak that falls below the peak threshold is also used to satisfy the peak excursion criteria. For example, a peak that is equal to 3 dB above the threshold when the peak excursion is equal to 6 dB will be found if the peak extends an additional 3 dB or more below the threshold level.

Type: float

```
instr.peak_excursion = 2
if instr.peak_excursion == 2:
   pass
```

#### HP856Xx.set\_marker\_to\_reference\_level()

Set the reference level to the amplitude of an active marker.

If no marker is active, 'marker\_to\_reference\_level' places a marker at the center of the trace and uses that marker amplitude to set the reference level.

#### HP856Xx.set\_marker\_delta\_to\_span()

Set the frequency span equal to the frequency difference between two markers on a trace.

The start frequency is set equal to the frequency of the left- most marker and the stop frequency is set equal to the frequency of the right-most marker.

### HP856Xx.set\_marker\_to\_center\_frequency\_step\_size()

Set the center frequency step-size equal to the frequency value of the active marker.

#### HP856Xx.marker\_time

Control the marker's time value. Default units are seconds.

Type: float

```
# set marker at sweep time corresponding second two
instr.marker_time = 2

if instr.marker_time == 2:
    pass
```

### HP856Xx.marker\_signal\_tracking\_enabled

Control whether the center frequency follows the active marker.

This is done after every sweep, thus maintaining the marker value at the center frequency. This allows you to "zoom in" quickly from a wide span to a narrow one, without losing the signal from the screen. Or, use 'marker\_signal\_tracking\_enabled' to keep a slowly drifting signal centered on the display. When this function is active, a "K" appears on the left edge of the display.

Type: bool

# **Diagnostic Values**

### HP856Xx.sampling\_frequency

Get the sampling oscillator frequency corresponding to the current start frequency. Diagnostic Attribute

Type: float

# HP856Xx.lo\_frequency

Get the first local oscillator frequency corresponding to the current start frequency. Diagnostic Attribute

Type: float

#### HP856Xx.mroll\_frequency

Get the main roller oscillator frequency corresponding to the current start frequency, except then the resolution bandwidth is less than or equal to 100 Hz.

Diagnostic Attribute

Type: float

# HP856Xx.oroll\_frequency

Get the offset roller oscillator frequency corresponding to the current start frequency, except when the resolution bandwidth is less than or equal to 100 Hz.

Diagnostic Attribute

Type: float

# HP856Xx.xroll\_frequency

Get the transfer roller oscillator frequency corresponding to the current start frequency, except when the resolution bandwidth is less than or equal to 100 Hz.

Diagnostic Attribute

Type: float

### HP856Xx.sampler\_harmonic\_number

Get the sampler harmonic number corresponding to the current start frequency.

Diagnostic Attribute

Type: int

#### Sweep

### HP856Xx.sweep\_single = <function HP856Xx.sweep\_single>

#### HP856Xx.sweep\_time

Control the sweep time. This is normally a coupled function which is automatically set to the optimum value allowed by the current instrument settings. Alternatively, you may specify the sweep time. Note that when the specified sweep time is too fast for the current instrument settings, the instrument is no longer calibrated and the message 'MEAS UNCAL' appears on the display. The sweep time cannot be adjusted when the resolution bandwidth is set to 10 Hz, 30 Hz, or 100 Hz.

Type: str, float

Real from 50E—3 to 100 when the span is greater than 0 Hz; 50E—6 to 60 when the span equals 0 Hz. When the resolution bandwidth is <100 Hz, the sweep time cannot be adjusted.

### HP856Xx.sweep\_couple

Control the sweep couple mode which is either a stimulus-response or spectrum-analyzer auto-coupled sweep time. In stimulus-response mode, auto-coupled sweep times are usually much faster for swept-response measurements. Stimulus-response auto-coupled sweep times are typically valid in stimulus-response measurements when the system's frequency span is less than 20 times the bandwidth of the device under test.

Type: str or SweepCoupleMode

### HP856Xx.sweep\_output

Control the sweep-related signal that is available from J8 on the rear panel. FAV provides a dc ramp of 0.5V/GHz. RAMP provides a 0—10 V ramp corresponding to the sweep ramp that tunes the first local oscillator (LO). For the HP 8561B, in multiband sweeps one ramp is provided for each frequency band.

Type: str or SweepOut

HP856Xx.set\_continuous\_sweep = <function HP856Xx.set\_continuous\_sweep>

### HP856Xx.trigger\_sweep()

Command the spectrum analyzer to take one full sweep across the trace display. Commands following TS are not executed until after the analyzer has finished the trace sweep. This ensures that the instrument is set to a known condition before subsequent commands are executed.

#### **Normalization**

### HP856Xx.normalize\_trace\_data\_enabled

Control the normalization routine for stimulus-response measurements. This function subtracts trace B from trace A, offsets the result by the value of the normalized reference position (normalized\_reference\_level), and displays the result in trace A. 'normalize\_trace\_data\_enabled' is intended for use with the store\_open() and store\_short() or store\_thru() commands. These functions are used to store a reference trace into trace B. Refer to the respective command descriptions for more information. Accurate normalization occurs only if the reference trace and the measured trace are on-screen. If any of these traces are off-screen, an error message will be displayed. If the error message ERR 903 A > DLMT is displayed, the range level (RL) can be adjusted to move the measured response within the displayed measurement range of the analyzer. If ERR 904 B > DLMT is displayed, the calibration is invalid and a thru or open/short calibration must be performed. If active (ON), the 'normalize\_trace\_data' command is automatically turned off with an instrument preset (IP) or at power on.

Type: bool

#### HP856Xx.normalized\_reference\_level

Control the normalized reference level. It is intended to be used with the normalize\_trace\_data command. When using 'normalized\_reference\_level', the input attenuator and IF step gains are not affected. This function is a trace-offset function enabling the user to offset the displayed trace without introducing hardware-switching errors into the stimulus-response measurement. The unit of measure for 'normalized\_reference\_level' is dB. In absolute power mode (dBm), reference level ( reference\_level) affects the gain and RF attenuation settings of the instrument, which affects the measurement or dynamic range. In normalized mode (relative power or dB-measurement mode), NRL offsets the trace data on-screen and does not affect the instrument gain or attenuation settings. This allows the displayed normalized trace to be moved without decreasing the measurement accuracy due to changes in gain or RF attenuation. If the measurement range must be changed to bring trace data on-screen, then the range level should be adjusted. Adjusting the range-level normalized mode has the same effect on the instrument settings as does reference level in absolute power mode (normalize off).

### Type: int

```
# reference level in case of normalization to -30 DB
instr.normalized_reference_level = -30

if instr.normalized_reference_level == -30:
    pass
```

### HP856Xx.normalized\_reference\_position

Control the normalized reference-position that corresponds to the position on the graticule where the difference between the measured and calibrated traces resides. The dB value of the normalized reference-position is equal to the normalized reference level. The normalized reference-position may be adjusted between 0.0 and 10.0, corresponding to the bottom and top graticule lines, respectively.

Type: float

```
instr.normalized_reference_position = 5.5
if instr.normalized_reference_position == 5.5:
    pass
```

### **Open/Short Calibration (Reflection)**

### HP856Xx.recall\_open\_short\_average()

Set the internally stored open/short average reference trace into trace B. The instrument state is also set to the stored open/short reference state.

```
instr.preset()
instr.sweep_single()
instr.start_frequency = 300e3
instr.stop_frequency = 1e9

instr.source_power_enabled = True
instr.sweep_couple = SweepCoupleMode.StimulusResponse
instr.source_peak_tracking()

input("CONNECT OPEN. PRESS CONTINUE WHEN READY TO STORE.")
instr.trigger_sweep()
instr.done()
instr.store_open()
```

(continues on next page)

(continued from previous page)

```
input("CONNECT SHORT. PRESS CONTINUE WHEN READY TO STORE AND AVERAGE.")
instr.trigger_sweep()
instr.done()
instr.store short()
input("RECONNECT DUT. PRESS CONTINUE WHEN READY.")
instr.trigger_sweep()
instr.done()
instr.normalize = True
instr.trigger_sweep()
instr.done()
instr.normalized_reference_position = 8
instr.trigger_sweep()
instr.preset()
# demonstrate recall of open/short average trace
instr.recall_open_short_average()
instr.trigger_sweep()
```

# HP856Xx.store\_open()

Save the current instrument state and trace A into nonvolatile memory.

This command must be used in conjunction with the <code>store\_short()</code> command and must precede the <code>store\_short()</code> command. The data obtained during the store open procedure is averaged with the data obtained during the <code>store\_short()</code> procedure to provide an open/short calibration. The instrument state (that is, instrument settings) must not change between the <code>store\_open()</code> and <code>store\_short()</code> operations in order for the open/short calibration to be valid. Refer to the <code>store\_short()</code> command description for more information.

### HP856Xx.store\_short()

Take currently displayed trace A data and averages this data with previously stored open data, and stores it in trace B.

This command is used in conjunction with the <code>store\_open()</code> command and must be preceded by it for proper operation. Refer to the <code>store\_open()</code> command description for more information. The state of the open/short average trace is stored in state register #8.

### **Thru Calibration**

# HP856Xx.store\_thru()

Store a thru-calibration trace into trace B and into the nonvolatile memory of the spectrum analyzer.

The state of the thru information is stored in state register #9.

#### HP856Xx.recall\_thru()

Recalls the internally stored thru-reference trace into trace B.

The instrument state is also set to the stored thru-reference state.

# **HP8560A Specific Attributes & Methods**

class pymeasure.instruments.hp.HP8560A(adapter, name='Hewlett-Packard HP8560A', \*\*kwargs)

Bases: HP856Xx

Represents the HP 8560A Spectrum Analyzer and provides a high-level interface for interacting with the instrument.

```
from pymeasure.instruments.hp import HP8560A
from pymeasure.instruments.hp.hp856Xx import AmplitudeUnits

sa = HP8560A("GPIB::1")

sa.amplitude_unit = AmplitudeUnits.DBUV
sa.start_frequency = 299.5e6
sa.stop_frequency = 300.5e6

print(sa.marker_amplitude)
```

### activate\_source\_peak\_tracking()

Activate a routine which automatically adjusts both the coarse and fine-tracking adjustments to obtain the peak response of the tracking generator on the spectrum-analyzer display. Tracking peak is not necessary for resolution bandwidths greater than or equal to 300 kHz. A thru connection should be made prior to peaking in order to ensure accuracy.



Only available with an HP 8560A Option 002.

## property source\_leveling\_control

Control if internal or external leveling is used with the built-in tracking generator. Takes either 'INT', 'EXT' or members of enumeration SourceLevelingControlMode

Type: str

```
instr.preset()
instr.sweep_single()
instr.center_frequency = 300e6
instr.span = 1e6

instr.source_power = -5

instr.trigger_sweep()
instr.source_leveling_control = SourceLevelingControlMode.External

if ErrorCode(900) in instr.errors:
    print("UNLEVELED CONDITION. CHECK LEVELING LOOP.")
```

### 1 Note

Only available with an HP 8560A Option 002.

#### property source\_power

Control the built-in tracking generator's output power.

Type: str, float



Only available with an HP 8560A Option 002.

#### property source\_power\_enabled

Set the built-in tracking generator on or off. See *source\_power* 

### property source\_power\_offset

Control the offset of the displayed power of the built-in tracking generator so that it is equal to the measured power at the input of the spectrum analyzer. This function may be used to take into account system losses (for example, cable loss) or gains (for example, preamplifier gain) reflecting the actual power delivered to the device under test.

Type: int

1 Note

Only available with an HP 8560A Option 002.

#### property source\_power\_step

Control the step size of the source power level, source power offset, and power-sweep range functions. Range: 0.1 ... 12.75 DB with 0.05 steps.

Type: float

1 Note

Only available with an HP 8560A Option 002.

### property source\_power\_sweep

Control the power-sweep function, where the output power of the tracking generator is swept over the power-sweep range chosen. The starting source power level is set using the *source\_power* command. The output power of the tracking generator is swept according to the sweep rate of the spectrum analyzer.

Type: str, float

1 Note

Only available with an HP 8560A Option 002.

### property source\_power\_sweep\_enabled

Set the power sweep active or inactive. See *source\_power\_sweep*.

# property tracking\_adjust\_coarse

Control the coarse adjustment to the frequency of the built-in tracking-generator oscillator. Once enabled,

this adjustment is made in digital-to-analogous or (DAC) values from 0 to 255. For fine adjustment, refer to the tracking\_adjust\_fine command description.

Type: int



#### Note

Only available with an HP 8560A Option 002.

### property tracking\_adjust\_fine

Control the fine adjustment of the frequency of the built-in tracking-generator oscillator. Once enabled, this adjustment is made in digital-to-analogounverter (DAC) values from 0 to 255. For coarse adjustment, refer to the tracking\_adjust\_coarse command description.

Type: int



#### Note

Only available with an HP 8560A Option 002.

# **HP8561B Specific Attributes & Methods**

class pymeasure.instruments.hp.HP8561B(adapter, name='Hewlett-Packard HP8561B', \*\*kwargs)

Bases: HP856Xx

Represents the HP 8561B Spectrum Analyzer and provides a high-level interface for interacting with the instrument.

```
from pymeasure.instruments.hp import 8561B
from pymeasure.instruments.hp.hp856Xx import AmplitudeUnits
sa = HP8560A("GPIB::1")
sa.amplitude_unit = AmplitudeUnits.DBUV
sa.start_frequency = 6.4e9
sa.stop\_frequency = 6.5e9
print(sa.marker_amplitude)
```

### property conversion\_loss

Control the compensation for losses outside the instrument when in external mixer mode (such as losses within connector cables, external mixers, etc.). 'conversion loss' specifies the mean conversion loss for the current harmonic band. In a full frequency band (such as band K), the mean conversion loss is defined as the minimum loss plus the maximum loss for that band divided by two. Adjusting for conversion loss allows the system to remain calibrated (that is, the displayed amplitude values have the conversion loss incorporated into them). The default value for any band is 30 dB. The spectrum analyzer must be in external-mixer mode in order for this command to work. When in internal-mixer mode, querying 'conversion loss' returns a zero.

#### property harmonic\_number\_lock

Control the lock to a chosen harmonic so only that harmonic is used to sweep an external frequency band. To select a frequency band, use the 'fullband' command; it selects an appropriate harmonic for the desired band. To change the harmonic number, use 'harmonic number lock'. Note that 'harmonic number lock'

also works in internal-mixing modes. Once 'fullband' or 'harmonic\_number\_lock' are set, only center frequencies and spans that fall within the frequency band of the current harmonic may be entered. When the 'set full span' command is activated, the span is limited to the frequency band of the selected harmonic.

### property harmonic\_number\_lock\_enabled

Set the harmonic number locking active or inactive. See <a href="harmonic\_number\_lock">harmonic\_number\_lock</a>.

### property mixer\_bias

Set the bias for an external mixer that requires diode bias for efficient mixer operation. The bias, which is provided on the center conductor of the IF input, is activated when MBIAS is executed. A "+" or "—" appears on the left edge of the spectrum analyzer display, indicating that positive or negative bias is on. When the bias is turned off, MBIAS is set to 0. Default units are in milliamps.

# property mixer\_bias\_enabled

Control the bias for an external mixer. See mixer\_bias.

### property mixer\_mode

Control the mixer mode. Select either the internal mixer or supply an external mixer. Takes enum 'Mixer-Mode' or string 'INT', 'EXT'

#### peak\_preselector()

Peaks the preselector in the HP 8561B Spectrum Analyzer.

Make sure the entire frequency span is in high band, set the desired trace to clear-write mode, place a marker on a desired signal, then execute PP. The peaking routine zooms to zero span, peaks the preselector tracking, then returns to the original position. To read the new preselector peaking number, use the PSDAC command. Commands following PP are not executed until after the analyzer has finished peaking the preselector.

### property preselector\_dac\_number

Control the preselector peak DAC number. For use with an HP 8561B Spectrum Analyzer.

Type: int

#### set\_fullband(band)

Select a commonly-used, external-mixer frequency band, as shown in the table. The harmonic lock function *harmonic\_number\_lock* is also set; this locks the harmonic of the chosen band. External-mixing functions are not available with an HP 8560A Option 002. Takes frequency band letter as string.

Table 2: Title

Frequency Band	Frequency (GHz)	Range	Mixing Harmonic	Conversion Loss
K	18.0 — 26.5		6	30 dB
A	26.5 - 40.0		8	30 dB
Q	33.0—50.0		10	30 dB
U	40.0—60.0		10	30 dB
V	50.0—75.0		14	30 dB
E	60.090.0		16	30 dB
W	75.0—110.0		18	30 dB
F	90.0—140.0		24	30 dB
D	110.0—170.0		30	30 dB
G	140.0—220.0		36	30 dB
Y	170.0—260.0		44	30 dB
J	220.0—325.0		54	30 dB

#### set\_signal\_identification\_to\_center\_frequency()

Set the center frequency to the frequency obtained from the command SIGID.

SIGID must be in AUTO mode and have found a valid result for this command to execute properly. Use SIGID on signals greater than 18 GHz {i.e., in external mixing mode). SIGID and IDCF may also be used on signals less than 6.5 GHz in an HP 8561B.

### property signal\_identification

Control the signal identification for identifying signals for the external mixing frequency bands. Two signal identification methods are available. AUTO employs the image response method for locating correct mixer responses. Place a marker on the desired signal, then activate signal\_identification = 'AUTO'. The frequency of a correct response appears in the active function block. Use this mode before executing the signal\_identification\_to\_center\_frequency() command. The second method of signal identification, 'MAN', shifts responses both horizontally and vertically. A correct response is shifted horizontally by less than 80 kHz. To ensure accuracy in MAN mode, limit the frequency span to less than 20 MHz. Where True = manual mode is active and False = auto mode is active or 'signal\_identification' is off.

### property signal\_identification\_frequency

Measure the frequency of the last identified signal. After an instrument preset or an invalid signal identification, IDFREQ returns a "0".

#### unlock\_harmonic\_number()

Unlock the harmonic number, allowing you to select frequencies and spans outside the range of the locked harmonic number.

Also, when HNUNLK is executed, more than one harmonic can then be used to sweep across a desired span. For example, sweep a span from 18 GHz to 40 GHz. In this case, the analyzer will automatically sweep first using 6—, then using 8—.

class pymeasure.instruments.hp.hp856Xx.AmplitudeUnits(value, names=<not given>, \*values,

#### **Enumerations**

```
Bases: StrEnum

Enumeration to represent the amplitude units.

AUTO = 'AUTO'
Automatic Unit (Usually derives to 'DBM')

DBM = 'DBM'
DB over millit Watt

DBMV = 'DBMV'
DB over milli Volt

DBUV = 'DBUV'
DB over micro Volt

MANUAL = 'MAN'
Manual Mode

V = 'V'
Volts
```

module=None, qualname=None, type=None,

*start=1*, *boundary=None*)

```
W = 'W'
```

Watt

Bases: StrEnum

Enumeration to represent the Mixer Mode of the HP8561B.

External = 'EXT'

Mixer Mode External

Internal = 'INT'

Mixer Mode Internal

Bases: StrEnum

Enumeration to represent either Trace A or Trace B.

A = 'TRA'

Trace A

B = 'TRB'

Trace B

Bases: StrEnum

Enumeration to represent the Coupling Mode.

AC = 'AC'

AC

DC = 'DC'

DC

Bases: StrEnum

Enumeration to represent the Demodulation Mode.

Amplitude = 'AM'

Amplitude Modulation

Frequency = 'FM'

Frequency Modulation

Off = 'OFF'

Demodulation Off

```
class pymeasure.instruments.hp.hp856Xx.DetectionModes(value, names=<not given>, *values,
                                                              module=None, qualname=None, type=None,
                                                              start=1, boundary=None)
     Bases: StrEnum
     Enumeration to represent the Detection Modes.
     NegativePeak = 'NEG'
          Negative Peak Detection
     Normal = 'NRM'
          Normal Peak Detection
     PositivePeak = 'POS'
          Positive Peak Detection
     Sample = 'SMP'
          Sampl Mode Detection
class pymeasure.instruments.hp.hp856Xx.ErrorCode(code)
     Bases: object
     Class to decode error codes from the spectrum analyzer.
class pymeasure.instruments.hp.hp856Xx.FrequencyReference(value, names=<not given>, *values,
                                                                   module=None, qualname=None,
                                                                   type=None, start=1, boundary=None)
     Bases: StrEnum
     Enumeration to represent the frequency reference source.
     External = 'EXT'
          External Frequency Standard
     Internal = 'INT'
          Internal Frequency Reference
class pymeasure.instruments.hp.hp856Xx.PeakSearchMode(value, names=<not given>, *values,
                                                              module=None, qualname=None, type=None,
                                                              start=1, boundary=None)
     Bases: StrEnum
     Enumeration to represent the Marker Peak Search Mode.
     High = 'HI'
          Place marker to the highest value on the trace
     NextHigh = 'NH'
          Place marker to the next highest value on the trace
     NextLeft = 'NL'
          Place marker to the next peak to the left
     NextRight = 'NR'
          Place marker to the next peak to the right
class pymeasure.instruments.hp.hp856Xx.SourceLevelingControlMode(value, names=<not given>,
                                                                           *values, module=None,
                                                                           qualname=None, type=None,
                                                                           start=1, boundary=None)
```

```
Bases: StrEnum
     Enumeration to represent the Source Leveling Control Mode of the HP8560A.
     External = 'EXT'
          Source Leveling Control Mode External
     Internal = 'INT'
          Source Leveling Control Mode Internal
class pymeasure.instruments.hp.hp856Xx.StatusRegister(value, names=<not given>, *values,
                                                              module=None, qualname=None, type=None,
                                                              start=1, boundary=None)
     Bases: IntFlag
     Enumeration to represent the Status Register.
     COMMAND\_COMPLETE = 16
          Any command is completed
     END_OF_SWEEP = 4
          Set when any sweep is completed
     ERROR_PRESENT = 32
          Set when error present
     MESSAGE = 2
          Set when display message appears
     NA = 8
          Unused but sometimes set
     NONE = 0
          No Interrupts can interrupt the program sequence
     RQS = 64
          Request Service
     TRIGGER = 1
          Trigger is activated
class pymeasure.instruments.hp.hp856Xx.SweepCoupleMode(value, names=<not given>, *values,
                                                                start=1, boundary=None)
     Bases: StrEnum
```

module=None, qualname=None, type=None,

Enumeration. SpectrumAnalyzer = 'SA' Stimulus Response

StimulusResponse = 'SR'

Spectrum Analyeze

class pymeasure.instruments.hp.hp856Xx.SweepOut(value, names=<not given>, \*values, module=None, *qualname=None*, *type=None*, *start=1*, boundary=None)

```
Bases: StrEnum
     Enumeration.
     Fav = 'FAV'
          DC Ramp 0.5V / GHz
     Ramp = 'RAMP'
          0 - 10V Ramp
class pymeasure.instruments.hp.hp856Xx.TriggerMode(value, names=<not given>, *values,
                                                           module=None, qualname=None, type=None,
                                                           start=1, boundary=None)
     Bases: StrEnum
     Enumeration to represent the different trigger modes
     External = 'EXT'
          External Mode
     Free = 'FREE'
          Free Running
     Line = 'LINE'
          Line Mode
     Video = 'VID'
          Video Mode
class pymeasure.instruments.hp.hp856Xx.WindowType(value, names=<not given>, *values,
                                                         module=None, qualname=None, type=None,
                                                         start=1, boundary=None)
     Bases: StrEnum
     Enumeration to represent the different window mode for FFT functions
     Flattop = 'FLATTOP'
          Flattop provides optimum amplitude accuracy
     Hanning = 'HANNING'
          Hanning provides an amplitude accuracy/frequency resolution compromise
     Uniform = 'UNIFORM'
          Uniform provides equal weighting of the time record for measuring transients.
```

# 7.28.7 HP Signal generator HP8657B

Note:

- This instrument does not support reading back values, as it is a listen-only GPIB device.
- Other instruments of this family could be implemented using the dynamic ranges feature.
- Optional pulse modulation feature is not supported yet.

### Glossary:

Abbreviation	Explanation	
AM	Amplitude Modulation	
FM	Frequency Modulation	
dBm	power level in dB referenced to 1mW	

class pymeasure.instruments.hp.HP8657B(adapter, name='Hewlett-Packard HP8657B', \*\*kwargs)

Bases: Instrument

Represents the Hewlett Packard 8657B signal generator and provides a high-level interface for interacting with the instrument.

**class Modulation**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntEnum

IntEnum for the different modulation sources

### property am\_depth

Set the modulation depth for AM, usable range 0-99.9%

### property am\_source

Set the source for the AM function with *Modulation* enumeration.

Value	Meaning
OFF	no modulation active
INT_400HZ	internal 400 Hz modulation source
INT_1000HZ	internal 1000 Hz modulation source
EXTERNAL	External source, AC coupling

# Note:

- AM & FM can be active at the same time
- only one internal source can be active at the time
- use "OFF" to deactivate AM

### usage example:

# check\_errors()

Method to read the error status register as the 8657B does not support any readout of values, this will return 0 and log a warning

### clear()

Reset the instrument to power-on default settings

### property fm\_deviation

Set the peak deviation in kHz for the FM function, useable range 0.1 - 400 kHz

#### NOTE:

the maximum usable deviation is depending on the output frequency, refer to the instrument documentation for further detail.

### property fm\_source

Set the source for the FM function with Modulation enumeration.

Value	Meaning
OFF	no modulation active
INT_400HZ	internal 400 Hz modulation source
INT_1000HZ	internal 1000 Hz modulation source
<b>EXTERNAL</b>	External source, AC coupling
DC_FM	External source, DC coupling (FM only)

#### Note:

- AM & FM can be active at the same time
- only one internal source can be active at the time
- use "OFF" to deactivate FM
- refer to the documentation rearding details on use of DC FM mode

### usage example:

#### property frequency

Set the output frequency of the instrument in Hz.

For the 8567B the valid range is 100 kHz to 2060 MHz.

#### id = 'HP.8657B.N/A.N/A'

Manual ID entry

### property level

Set the output level in dBm.

For the 8657B the range is -143.5 to +17 dBm/

### property level\_offset

Set the output offset in dB, usable range -199 to +199 dB.

#### property output\_enabled

Control whether the output is enabled.

### reset()

Resets the instrument.

### shutdown()

Brings the instrument to a safe and stable state

# 7.28.8 HP 8753E Vector Network Analyzer

Bases: Instrument

Represents the HP8753E Vector Network Analyzer and provides a high-level interface for taking sweeps of the scattering or measurement parameters.

#### **Parameters**

- name Optional set the name of the instrument (str).
- min\_frequency Optional set the minimum validator for HP8753E.start\_frequency, HP8753E.stop\_frequency, HP8753E.span\_frequency, HP8753E.center\_frequency on initialization (float in Hz).
- max\_frequency Optional set the maximum validator for HP8753E.start\_frequency, HP8753E.stop\_frequency, HP8753E.stop\_frequency, HP8753E.stop\_frequency on initialization (float in Hz).
- **min\_power** Optional set the minimum validator for *HP8753E.power* on initialization (float in dBm).
- max\_power Optional set the maximum validator for *HP8753E.power* on initialization (float in dBm).

#### property IFBW

Control the IF Bandwidth of the instrument for a scan sweep. (int from the set [10, 30, 100, 300, 1000, 3000, 3700, 6000]).

### property averages

Control the number of averages for a scan sweep. (int truncated from 1 to 999).

### property averaging\_enabled

Control whether or not averaging is enabled. (boolean)

### averaging\_restart()

Restart sweep averaging.

### property center\_frequency

Control the center frequency in Hz (float).(dynamic)

#### property correction\_enabled

Control whether or not correction is enabled. (boolean)

#### property data\_complex

Get the complex s-parameter measurements from the last scan. This function is blocking until it is completed.

#### **Parameters**

**timeout** – Optional value in seconds to wait until a timeout occurs.

#### Raises

**TimeoutError** – If the VNA fails to complete the sweep before timeout occurs.

#### Returns

An array of s-parameters for the measurement\_parameter.

### **Return type**

numpy.ndarray

### emit\_beep()

Make the VNA emit a beep

# property frequencies

Get a list of frequencies from the last scan.

#### Returns

An array of frequencies sized by number of points in sweep)

#### Return type

numpy.ndarray

### property fw

Get the firmware of the instrument.

### property id

Get the identification of the instrument (list of strings)

### property manu

Get the manufacturer of the instrument.

# property measuring\_parameter

Get the active Scattering or Measuring Parameter being measured. (str from ['S11', 'S21', 'S12', 'S22', 'A', 'B', 'R'])

### property model

Get the model of the instrument.

#### property options

Get the installed options for the instrument

### property output\_enabled

Control RF Output State (boolean)

# property power

Control the output RF power of the instrument's active port. (float).(dynamic)

### reset()

Reset the instrument. May cause RF Output power to be enabled!

### scan(timeout=10)

Initiates a scan with the number of averages specified and blocks until the operation is complete.

### **Parameters**

**timeout** – Optional value in seconds to add to the sweep time before timeout occurs.

#### Raises

**TimeoutError** – If the VNA fails to complete the sweep before timeout occurs.

#### Returns

None.

### property scan\_points

Control the number of points used for a scan sweep. From the set [3, 11, 21, 26, 51, 101, 201, 401, 801, 1601]

### scan\_single()

Initiates a single scan or N scans averaged based on averaging This function is not blocking.

### set\_fixed\_frequency(frequency)

Set the sweep to be a fixed frequency in Hz.

#### set\_sweep\_time\_fastest()

Set instrument scan sweep time to select fastest possible time.

#### shutdown()

Shutdown - Disables RF Output.

#### property sn

Get the serial number for the instrument

#### property span\_frequency

Control the span of the sweep frequency in Hz (float).(dynamic)

#### property start\_frequency

Control the start frequency in Hz. (float).(dynamic)

### property stop\_frequency

Control the stop frequency in Hz. (float).(dynamic)

### property sweep\_time

Control the sweep time in seconds. (float truncated from 0.0 to 36\_400.0)

#### property trigger\_continuous

Control Sweep Scan Trigger State (boolean)

# property trigger\_hold

Control Sweep Scan Trigger State (boolean)

# 7.28.9 HP 11713A Attenuator/Switch Driver

```
class pymeasure.instruments.hp.HP11713A(adapter, name='Hewlett-Packard HP11713A', **kwargs)
```

Bases: Instrument

Represents the HP 11713A Switch and Attenuator Driver and provides a high-level interface for interacting with the instrument.

Usually an attenuator is hooked to either X or Y or X and Y. To ease the control of the attenuator driver you have the possibility to set an attenuator type via the attribute 'ATTENUATOR\_X' or 'ATTENUATOR\_Y'. The hp11713a keeps different default attenuator mappings. After setting the attenuator type you are able to use the methods 'attenuation\_x' and/or 'attenuation\_y' to set the switch driver to the correct value for the specified attenuation. The attenuation values are rounded.

```
from pymeasure.instruments.hp import HP11713A
from pymeasure.instruments.hp.hp11713a import Attenuator_110dB

sd = HP11713A("GPIB::1")

sd.ATTENUATOR_Y = Attenuator_110dB
sd.attenuation_y(10)
sd.ch_0.enabled = True
```

#### channels

#### Channels

```
ch_0: SwitchDriverChannel, ch_1: SwitchDriverChannel, ch_2:
SwitchDriverChannel, ch_3: SwitchDriverChannel, ch_4: SwitchDriverChannel,
ch_5: SwitchDriverChannel, ch_6: SwitchDriverChannel, ch_7:
SwitchDriverChannel, ch_8: SwitchDriverChannel
```

#### attenuation\_x(attenuation)

Set switches according to the attenuation in dB for X

The set attenuation will be rounded to the next available step. An attenuation mapping has to be set in before e.g.

```
from pymeasure.instruments.hp.hp11713a import HP11713A, Attenuator_110dB
instr.ATTENUATOR_X = Attenuator_110dB
instr.attenuation_x(10)
```

#### attenuation\_y(attenuation)

Set switches according to the attenuation in dB for Y

The set attenuation will be rounded to the next available step. An attenuation mapping has to be set in before e.g.

```
from pymeasure.instruments.hp.hp11713a import HP11713A, Attenuator_110dB
instr.ATTENUATOR_Y = Attenuator_110dB
instr.attenuation_y(10)
```

### deactivate\_all()

Deactivate all switches to polarity 'B'.

class pymeasure.instruments.hp.hp11713a.SwitchDriverChannel(parent, id)

Bases: Channel

### property enabled

Set this channel to the polarity 'A' for True and 'B' for False.

```
pymeasure.instruments.hp.hp11713a.Attenuator_11dB
```

Mapping of logical values for use with 0 - 11 dB attenuators

```
pymeasure.instruments.hp.hp11713a.Attenuator_110dB
```

Mapping of logical values for use with 0 - 110 dB attenuators

```
pymeasure.instruments.hp.hp11713a.Attenuator_70dB_3_Section
```

Mapping of logical values for use with 0 - 70 dB attenuators with 3 switching sections

#### pymeasure.instruments.hp.hp11713a.Attenuator\_70dB\_4\_Section

Mapping of logical values for use with 0 - 70 dB attenuators with 4 switching sections

### 7.28.10 HP 437B Power Meter

class pymeasure.instruments.hp.HP437B(adapter, name='Hewlett-Packard HP437B', \*\*kwargs)

Bases: Instrument

Represents the HP437B Power Meters.



#### Note

Most command descriptions are taken from the document: 'Operating Manual 437B Power Meter'

### activate\_auto\_range()

The power meter divides each sensor's power range into 5 ranges of 10 dB each. Range 1 is the most sensitive (lowest power levels), and Range 5 is the least sensitive (highest power levels). Range 5 can be less than 10 dB if the sensor's power range is less than 50 dB. The range can be set either automatically or manually. 'activate\_auto\_range' automatically selects the correct range for the current measurement.

#### property automatic\_range\_enabled

Control the automatic range. The power meter divides each sensor's power range into 5 ranges of 10 dB each. Range 1 is the most sensitive (lowest power levels), and Range 5 is the least sensitive (highest power levels). The range can be set either automatically or manually.

### calibrate(calibration\_factor)

Calibrate a sensor to the power meter with a 'calibration factor' in percent.

#### property calibration\_factor

Control the calibration factor of a specific power sensor at a specific input frequency. (A chart or table of CAL FACTOR % versus Frequency is printed on each sensor and an accompanying data sheet.) Calibration factor is entered in percent. Valid entries for 'calibration factor' range from 1.0 to 150.0%.

#### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

### property display\_all\_segments\_enabled

Set all segments of the display of the power meter active or resume normal state.

### property display\_enabled

Set the display of the power meter active or inactive.

### property display\_output

Get the current displayed string of values of the power meter.

```
print(instr.display_output)
-0.23 dB REL
```

### property display\_user\_message

Set a custom user message up to 12 alpha-numerical chars. If the string is empty or None the user message gets disabled.

#### property duty\_cycle

Control the duty cycle for calculation of a pulsed input signal. This function will cause the power meter to report the pulse power of a rectangular pulsed input signal. The allowable range of values for 'duty\_cycle' is 0.00001 to 0.99999.

Pulse power, as reported by the power meter, is a mathematical representation of the pulse power rather than an actual measurement. The power meter measures the average power of the pulsed input signal and then divides the measurement by the duty cycle value to obtain a pulse power reading.

# property duty\_cycle\_enabled

Control whether the duty cycle is active or inactive. See duty\_cycle

#### property event\_status

Get the status byte and Master Summary Status bit.

```
print(instr.request_service_conditions)
StatusRegister.PowerOn|CommandError
```

### property filter

Control the filter number for averaging. Setting a value implicitly enables the manual filter mode. Setting a value of 1 basically disables the averaging.

### property filter\_automatic\_enabled

Control the filter mode. By switching over from automatic to manual (true to false) the instrument implicitly keeps (holds) the filter value from the automatic selection.

### property frequency

Control the frequency of the input signal. Entering a frequency causes the power meter to select a sensor-specific calibration factor. The allowed range of 'frequency' values is from 0.0001 to 999.9999 GHz with a 100 kHz resolution. The unit is Hz.

### property group\_trigger\_mode

Control the group execute trigger mode. When in remote and addressed to listen, the power meter responds to a Trigger message (the Group Execute Trigger bus command [GET]) according to the programmed mode.

#### property limit\_high

Control the upper limit for the builtin limit checking.

### property limit\_high\_hit

Get if the upper limit check got triggered.

#### property limit\_low

Control the lower limit for the builtin limit checking.

#### property limit\_low\_hit

Get if the lower limit check got triggered.

### property limits\_enabled

Control the limits checking function to allow the power meter to monitor the power level at the sensor and to indicate when that power is outside preset limits.

#### property linear\_display\_enabled

Control if the power meter displays or reports the power values in logarithmic or linear units. Set *linear\_display\_enabled* to 'True' to activate linear value readout.

```
from pymeasure.instruments.hp.hp437b import LogLin
instr.relative_mode_enabled = False
instr.linear_display_enabled = True
```

#### property measurement\_unit

Get the measurement unit the power meter is currently reporting the power values in.

Depends on: relative\_mode\_enabled and attr:linear\_display\_enabled

```
instr.relative_mode_enabled = False
instr.linear_display_enabled = True

print(instr.measurement_unit)
MeasurementUnit.Watts
```

#### property offset

Control the offset applied to the measured value to compensate for signal gain or loss (for example, to compensate for the loss of a 10 dB directional coupler). Offsets are entered in dB. In case the *offset\_enabled* is false this returns automatically 0.0

### property offset\_enabled

Control the offset being applied.

### property operating\_mode

Get the operating mode the power meter is currently in.

### property power

Measure the power at the power sensor attached to the power meter in the corresponding unit. In case a measurement would be invalid the power meter responds with the value float ('nan').

### property power\_reference\_enabled

Control the builtin reference power source 1mW @ 50 MHz.

## preset()

Sets the power meter to a known state. Preset conditions are shown in the following table.

Table 3: Preset values

Parameter	Value/Condition
Frequency	50 MHz
Resolution	0.01 dB
Duty Cylce	1.000%, Off
Relative	0 dB, Off
Power Reference	Off
Range	Auto
Unit	dBm
Low Limit	-90.000 dBm
High Limit	+90.000 dBm
Limit Checking	Off
Trigger Mode	Free Run
Group Trigger Mode	Trigger with Delay
Display Function	Display Enable

#### property range

Control the range to be selected manually. Valid range numbers are 1 through 5. See *automatic\_range\_enabled* for further information.

#### property relative\_mode\_enabled

Control the relative mode. In the relative mode the current measured power value will be used as reference and any further reported value from *power* will refer to this.

#### reset()

Resets the instrument.

# property resolution

Control the resolution of the power meter's measured value. Three levels of resolution can be set: 0.1 dB, 0.01 dB and 0.001 dB or if the selected unit is Watts 1%, 0.1% and 0.001%.

### sensor\_data\_clear(sensor\_id)

Clear the Sensor Data table of 'sensor\_id' previous to entering new values.

### sensor\_data\_read\_cal\_factor\_table(sensor\_id)

Read the Sensor Data calibration table. See <code>sensor\_data\_write\_cal\_factor\_table()</code> Returns a tuple of frequencies as list and calibration factors as list.

```
sensor_data_ref_cal_factor(sensor_id, ref_cal_factor)
```

Set the power sensor's reference calibration factor to the Sensor Data table.

```
sensor_data_write_cal_factor_table(sensor_id, frequency_table, cal_fac_table)
```

Write the 'calibration\_table' for 'sensor\_id' to the Sensor Data table. And write the reference calibration factor for the 'sensor\_id'. Frequency is given in Hz. Calibration factor as percentage.

The power meter's memory contains space for 10 tables, numbered 0—9. Tables 0-7 each contain space for 40 frequency/calibration factor pairs. Tables 8 and 9 each contain space for 80 frequency/calibration factor pairs.

This function clears the sensor table before writing.

Example table:

```
calibration_table = {
    10e6: 100.0,
    1e9: 96.5,
    2e9: 97.0
}
instr.sensor_data_cal_factor_table(0, calibration_table.keys(),
    calibration_table.values())
```

#### sensor\_data\_write\_id\_label(sensor\_id, label)

Set a particular power sensor's ID label table to be modified. The sensor ID label must not exceed 7 characters. For example, to identify Sensor Data table #2 with an ID number of 1234567:

```
instr.sensor_data_id_label(2, "1234567")
```

### property sensor\_type

Set the sensor type connected to the power meter to select the corresponding calibration factor.

```
from pymeasure.instruments.hp.hp437b import SensorType
instr.sensor_type = SensorType.HP_8481A
```

### store(register)

The power meter can store instrument configurations for recall at a later time. The following information can be stored in the power meter's internal registers:

- · reference calibration factor value
- Measurement units (dBm or watts)
- relative value and status (on or off)
- power reference status (on or off)
- · calibration factor value
- SENSOR ID (sensor data table selection)
- offset value and status (on or off)
- range (Auto or Set)
- frequency value
- · resolution
- duty cycle value and status (on or off)
- Filter (number of readings averaged, auto or manual)
- Limits value and status (on or off)

Registers 1 through 10 are available for storing instrument configurations.

# trigger\_delay()

Trigger with delay.

Triggering with delay is identical to <code>trigger\_immediate()</code> except the power meter inserts a settling-time delay before taking the requested measurement. This settling time allows the internal digital filter to be updated with new values to produce valid, accurate measurement results. The trigger with delay command allows time for settling of the internal amplifiers and filters. It does not allow time for power sensor delay. In cases of large power changes, the delay may not be sufficient for complete settling. Accurate readings can be assured by taking two successive measurements for comparison. Once the measurement results are displayed and read onto the bus, the power meter reverts to standby mode.

#### trigger\_immediate()

Trigger immediate.

When the power meter receives the trigger immediate program code, it inputs one more data point into the digital filter, measures the reading from the filter, and then updates the display and HP-IB. (When the trigger immediate command is executed, the internal digital filter is not cleared.) The power meter then waits for the measurement results to be read by the controller. While waiting, the power meter can process most bus commands without losing the measurement results. If the power meter receives a trigger immediate command and then receives the GET (Group Execute Trigger) command, the trigger immediate command will be aborted and a new measurement cycle will be executed. Once the measurement results are read onto the bus, the power meter always reverts to standby/hold mode. Measurement results obtained via trigger immediate are normally valid only when the power meter is in a steady, settled state.

#### property trigger\_mode

Control the trigger mode.

The power meter has two modes of triggered operation; standby mode and free run mode. Standby mode means the power meter is making measurements, but the display and HP-IB are not updated until a trigger command is received. Free run means that Meter takes measurements and updates the display and HP-IB continuously.

#### zero()

Adjust the power meter's internal circuitry for a zero power indication when no power is applied to the sensor.



#### 1 Note

Ensure that no power is applied to the sensor while the power meter is zeroing. Any applied RF input power will cause an erroneous reading.

# 7.28.11 Support class for HP legacy devices

Currently this implementation is used for the following instruments which do not support SCPI:

- HP3437A System-Voltmeter
- HP3478A Digital Multimeter
- HP6632/33/34A System power supply

class pymeasure.instruments.hp.HPLegacyInstrument(adapter, name='HP legacy instrument', \*\*kwargs)

Bases: Instrument

Class for legacy HP instruments from the era before SPCI, based on pymeasure. Instrument

### GPIB\_trigger()

Initate trigger via low-level GPIB-command (aka GET - group execute trigger)

# reset()

Initatiates a reset (like a power-on reset) of the HP3478A

#### shutdown()

provides a way to gracefully close the connection to the HP3478A

#### property status

Get an object representing the current status of the unit.

# status\_desc

alias of StatusBitsBase

# values(command, \*\*kwargs)

Write a command to the instrument and return a list of formatted values from the result.

## **Parameters**

- **command** SCPI command to be sent to the instrument.
- preprocess\_reply Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.

- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- **\*\*kwargs** Keyword arguments to be passed to the ask() method.

#### **Returns**

A list of the desired type, or strings where the casting fails.

### write(command)

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- command command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# 7.28.12 HP System Power Supplies HP663XA

Currently supported models are:

Model	Voltage	Current	Power
6632A	020 V	05.0 A	100 W
6633A	050 V	02.5 A	100 W
6634A	0100 V	01.0 A	100 W

#### Note:

- The multi-channel system power supplies HP 6621A, 6622A, 6623A, 6624A, 6625A, 6626A, 6627A & 6628A share some of the command syntax and could probably be incorporated in this implementation
- The B-version of these models (6632B, 6633B & 6634B) are SPCI-compliant and could be implemented in a similar manner

class pymeasure.instruments.hp.HP6632A(adapter, name='Hewlett-Packard HP6632A', \*\*kwargs)

Bases: HPLegacyInstrument

Represents the Hewlett Packard 6632A system power supply and provides a high-level interface for interacting with the instrument.

**class ERRORS**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: Enum

Enum class for error messages

# property OCP\_enabled

A bool property which controls if the OCP (OverCurrent Protection) is enabled

# property SRQ\_enabled

A bool property which controls if the SRQ (ServiceReQuest) is enabled

**class ST\_ERRORS**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: Enum

Enum class for selftest errors

### check\_errors()

Method to read the error status register

#### Return error status

one byte with the error status register content

### Rtype error\_status

int

#### check\_selftest\_errors()

Method to read the error status register

### Return error\_status

one byte with the error status register content

#### Rtype error\_status

int

#### clear()

Resets the instrument to power-on default settings

### property current

A floating point property that controls the output current of the device.

(dynamic)

### property delay

A float property that changes the reprogamming delay Default values: 8 ms in FAST mode 80 ms in NORM mode

Values will be rounded to the next 4 ms by the instrument

# property display\_active

A boot property which controls if the display is enabled

### property id

Reads the ID of the instrument and returns this value for now

#### property output\_enabled

A bool property which controls if the output is enabled

### property over\_voltage\_limit

A floationg point property that sets the OVP threshold.

(dynamic)

### reset\_OVP\_OCP()

Resets Overvoltage and Overcurrent protections

## property rom\_version

Reads the ROM id (software version) of the instrument and returns this value for now

### property status

Returns an object representing the current status of the unit.

### status\_desc

alias of Status

### property voltage

A floating point proptery that controls the output voltage of the device.

(dynamic)

class pymeasure.instruments.hp.HP6633A(adapter, name='Hewlett Packard HP6633A', \*\*kwargs)

Bases: HP6632A

Represents the Hewlett Packard 6633A system power supply and provides a high-level interface for interacting with the instrument.

class pymeasure.instruments.hp.HP6634A(adapter, name='Hewlett Packard HP6634A', \*\*kwargs)

Bases: HP6632A

Represents the Hewlett Packard 6634A system power supply and provides a high-level interface for interacting with the instrument.

## 7.29 Inficon

This section contains specific documentation on the Inficon instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.29.1 Inficon SQM-160 multi-film rate/thickness monitor

Bases: Instrument

Inficon SQM-160 multi-film rate/thickness monitor.

Uses a quartz crystal sensor to measure rate and thickness in a thin film deposition process. Connection to the device is commonly made through a serial connection (RS232) or optionally via USB or Ethernet.

## A command packet always consists of the following:

- 1 Byte: Sync character ('!' appears only at the start of a message).
- $1\ \mathrm{Byte}$ : length character obtained from the message length without CRC.

A value of 34 is added so that no '!' can occur.

- Command message with variable length.
- 2 Byte: Cyclic Redundancy Check (CRC) checksum.

## A response packet always consists of:

- 1 Byte: Sync character ('!' appears only at the start of a message).
- 1 Byte: length character obtained from the message length without CRC. A value of 35 is added.
- 1 Byte: Response status character indicating the status of the command.
- Response message with variable length.
- 2 Byte: Cyclic Redundancy Check (CRC) checksum.

### **Parameters**

• adapter – pyvisa resource name of the instrument or adapter instance

7.29. Inficon 319

- name (string) Name of the instrument.
- baud\_rate (string) Baud rate used by the serial connection.
- **kwargs** Any valid key-word argument for Instrument

#### sensor\_1

#### Channel

SensorChannel

#### sensor\_2

#### Channel

SensorChannel

#### sensor\_3

#### Channel

SensorChannel

### sensor\_4

#### Channel

SensorChannel

## sensor\_5

#### Channel

SensorChannel

### sensor\_6

## Channel

SensorChannel

## property all\_values

Get the current rate (Angstrom/s), Thickness (Angstrom), and frequency (Hz) for each sensor

#### property average\_rate

Get the current average rate in Angstrom per second

## property average\_thickness

Get the current average thickness in Angstrom

## check\_set\_errors()

Check the errors after setting a property.

## property firmware\_version

Get the firmware version.

## property number\_of\_channels

Get the number of installed channels

### read()

Reads a response message from the instrument.

This method also checks for a correct checksum.

### Returns

the response packet

### Return type

string

#### Raises

**ConnectionError** – if a checksum error is detected or a wrong response status is detected.

## property reset\_flag

Get the power-up reset flag. It is True only when read first after a power cycle.

#### reset\_system\_parameters()

Reset all film and system parameters.

## reset\_thickness\_rate()

Reset the average thickness and rate.

This also sets all active Sensor Rates and Thicknesses to zero

#### reset\_time()

Reset the time of the monitor to zero.

#### write(command)

Write a command to the device.

## class pymeasure.instruments.inficon.sqm160.SensorChannel(parent, id)

Bases: Channel

Sensor channel for individual rate measurements.

#### property crystal\_life

Get the crystal life value in percent

## property frequency

Get the current frequency for a sensor in Hz

## property rate

Get the current rate for a sensor in Angstrom per second

#### property thickness

Get the current thickness for a sensor in Angstrom

## 7.30 IPG Photonics

This section contains specific documentation on the IPG Photonics instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.30.1 YAR fiber amplifier series

**class** pymeasure.instruments.ipgphotonics.yar.**YAR**(adapter, name='YAR fiber amplifier', \*\*kwargs)

Bases: Instrument

Communication with the YAR fiber amplifier series by IPG Photonics.

This is the RS232 command set. GPIB has different commands.

7.30. IPG Photonics 321

**class Status**(value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntFlag

## check\_set\_errors()

Check for errors after having set a property.

Called if check\_set\_errors=True is set for that property.

## clear()

Reset all errors.

## property current

Measure the diode current in A.

## property emission\_enabled

Control emission of the amplifier (bool).

## property firmware

Get firmware version

## property id

Get the model number.

### property maximum\_case\_temperature

Measure the maximum temperature for the optical module in °C.

## property minimum\_display\_power

Measure the minimum displayable output power in W.

## property power

Measure current output power in W.(dynamic)

## property power\_range

Get the power limits in W.

### property power\_setpoint

Control output power setpoint in W.(dynamic)

#### read()

Read an instrument answer and check whether it is an error.

## property status

Get the current status.

#### property temperature

Measure case temperature in °C.

## property temperature\_seed

Measure current seed temperature in °C

## property wavelength\_temperature

Control temperature in °C for seed wavelength control.

# 7.31 Keithley

This section contains specific documentation on the Keithley instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.31.1 Keithley 2000 Multimeter

Bases: KeithleyBuffer, SCPIUnknownMixin, Instrument

Represents the Keithley 2000 Multimeter and provides a high-level interface for interacting with the instrument.

```
meter = Keithley2000("GPIB::1")
meter.measure_voltage()
print(meter.voltage)
```

## acquire\_reference(mode=None)

Sets the active value as the reference for the active mode, or can set another mode by its name.

#### **Parameters**

mode – A valid mode name, or None for the active mode

#### auto\_range(mode=None)

Sets the active mode to use auto-range, or can set another mode by its name.

#### Parameters

mode – A valid mode name, or None for the active mode

beep(frequency, duration)

Sounds a system beep.

#### **Parameters**

- **frequency** A frequency in Hz between 65 Hz and 2 MHz
- duration A time in seconds between 0 and 7.9 seconds

## property beep\_state

Control whether the system status beeper is enabled, which can take the values: enabled and disabled.

### property buffer\_data

Get a numpy array of values from the buffer.

#### property buffer\_points

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

## check\_errors()

Read all errors from the instrument.

#### **Returns**

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### **Returns**

List of error entries.

## clear()

Clear the instrument status byte.

## property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

## **Parameters**

- points The number of points in the buffer.
- **delay** The delay time in seconds.

#### property current

Get a DC or AC current measurement in Amps, based on the active *mode*.

## property current\_ac\_bandwidth

Control (floating) the AC current detector bandwidth in Hz, which can take the values 3, 30, and 300 Hz.

#### property current\_ac\_digits

Control (integer) the number of digits in the AC current readings, which can take values from 4 to 7.

#### property current\_ac\_nplc

Control (floating) the number of power line cycles (NPLC) for the AC current measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

## property current\_ac\_range

Control (floating) the AC current range in Amps, which can take values from 0 to 3.1 A. Auto-range is disabled when this property is set.

## property current\_ac\_reference

Control (floating) the AC current reference value in Amps, which can take values from -3.1 to 3.1 A.

## property current\_digits

Control (integer) the number of digits in the DC current readings, which can take values from 4 to 7.

## property current\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC current measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

#### property current\_range

Control (floating) the DC current range in Amps, which can take values from 0 to 3.1 A. Auto-range is disabled when this property is set.

## property current\_reference

Control (floating) the DC current reference value in Amps, which can take values from -3.1 to 3.1 A.

## disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

#### disable\_filter(mode=None)

Disables the averaging filter for the active mode, or can set another mode by its name.

#### **Parameters**

**mode** – A valid *mode* name, or None for the active mode

## disable\_reference(mode=None)

Disables the reference for the active mode, or can set another mode by its name.

#### Parameters

mode – A valid mode name, or None for the active mode

## enable\_filter(mode=None, type='repeat', count=1)

Enables the averaging filter for the active mode, or can set another mode by its name.

#### **Parameters**

- mode A valid mode name, or None for the active mode
- **type** The type of averaging filter, either 'repeat' or 'moving'.
- count A number of averages, which can take take values from 1 to 100

## enable\_reference(mode=None)

Enables the reference for the active mode, or can set another mode by its name.

#### **Parameters**

**mode** – A valid *mode* name, or None for the active mode

#### property frequency

Get a frequency measurement in Hz, based on the active *mode*.

## property frequency\_aperature

Control (floating) the frequency aperature in seconds, which sets the integration period and measurement speed. Takes values from 0.01 to 1.0 s.

#### property frequency\_digits

Control (integer) the number of digits in the frequency readings, which can take values from 4 to 7.

## property frequency\_reference

Control (floating) the frequency reference value in Hz, which can take values from 0 to 15 MHz.

### property frequency\_threshold

Control (floating) the voltage signal threshold level in Volts for the frequency measurement, which can take values from 0 to 1010 V.

## property id

Get the identification of the instrument.

#### is buffer full()

Return True if the buffer is full of measurements.

#### local()

Returns control to the instrument panel, and enables the panel if disabled.

### measure\_continuity()

Configures the instrument to perform continuity testing.

## measure\_current(max\_current=0.01, ac=False)

Configures the instrument to measure current, based on a maximum current to set the range, and a boolean flag to determine if DC or AC is required.

#### **Parameters**

- max\_current A current in Volts to set the current range
- ac False for DC current, and True for AC current

#### measure\_diode()

Configures the instrument to perform diode testing.

#### measure\_frequency()

Configures the instrument to measure the frequency.

## measure\_period()

Configures the instrument to measure the period.

#### measure\_resistance(max resistance=10000000.0, wires=2)

Configures the instrument to measure resistance, based on a maximum resistance to set the range, and the number of wires utilized.

#### **Parameters**

- max\_resistance A resistance in ohms to set the resistance range
- wires Number of wires employed (2 or 4)

#### measure\_temperature()

Configures the instrument to measure the temperature.

## measure\_voltage(max\_voltage=1, ac=False)

Configures the instrument to measure voltage, based on a maximum voltage to set the range, and a boolean flag to determine if DC or AC is required.

#### **Parameters**

- max\_voltage A voltage in Volts to set the voltage range
- ac False for DC voltage, and True for AC voltage

#### property mode

Control (string) the configuration mode for measurements, which can take the values: current (DC), current ac, voltage (DC), voltage ac, resistance (2-wire), resistance 4W (4-wire), period, temperature, diode, and frequency.

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

## property options

Get the device options installed.

#### property period

Get a period measurement in seconds, based on the active *mode*.

### property period\_aperature

Control (floating) the period aperature in seconds, which sets the integration period and measurement speed. Takes values from 0.01 to 1.0 s.

## property period\_digits

Control (integer) the number of digits in the period readings, which can take values from 4 to 7.

### property period\_reference

Control (floating) the period reference value in seconds, which can take values from 0 to 1 s.

## property period\_threshold

Control (floating) the voltage signal threshold level in Volts for the period measurement, which can take values from 0 to 1010 V.

### read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

## **Returns bytes**

Bytes response of the instrument (including termination).

### remote()

Places the instrument in the remote state, which is does not need to be explicitly called in general.

## remote\_lock()

Disables and locks the front panel controls to prevent changes during remote operations. This is disabled by calling *local()*.

#### reset(

Resets the instrument state.

## reset\_buffer()

Reset the buffer.

## property resistance

Get a resistance measurement in Ohms for both 2-wire and 4-wire configurations, based on the active mode.

### property resistance\_4W\_digits

Control (integer) the number of digits in the 4-wire resistance readings, which can take values from 4 to 7.

## property resistance\_4W\_nplc

Control (floating) the number of power line cycles (NPLC) for the 4-wire resistance measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property resistance\_4W\_range

Control (floating) the 4-wire resistance range in Ohms, which can take values from 0 to 120 MOhms. Auto-range is disabled when this property is set.

### property resistance\_4W\_reference

Control (floating) the 4-wire resistance reference value in Ohms, which can take values from 0 to 120 MOhms.

## property resistance\_digits

Control (integer) the number of digits in the 2-wire resistance readings, which can take values from 4 to 7.

## property resistance\_nplc

Control (floating) the number of power line cycles (NPLC) for the 2-wire resistance measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

#### property resistance\_range

Control (floating) the 2-wire resistance range in Ohms, which can take values from 0 to 120 MOhms. Auto-range is disabled when this property is set.

### property resistance\_reference

Control (floating) the 2-wire resistance reference value in Ohms, which can take values from 0 to 120 MOhms.

#### shutdown()

Brings the instrument to a safe and stable state

#### start\_buffer()

Starts the buffer.

## property status

Get the status byte and Master Summary Status bit.

#### stop buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

#### property temperature

Get a temperature measurement in Celsius, based on the active *mode*.

## property temperature\_digits

Control (integer) the number of digits in the temperature readings, which can take values from 4 to 7.

## property temperature\_nplc

Control (floating) the number of power line cycles (NPLC) for the temperature measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property temperature\_reference

Control (floating) the temperature reference value in Celsius, which can take values from -200 to 1372 C.

## property trigger\_count

Control (integer) the trigger count, which can take values from 1 to 9,999.

### property trigger\_delay

Control (floating) the trigger delay in seconds, which can take values from 1 to 9,999,999.999 s.

### property voltage

Get a DC or AC voltage measurement in Volts, based on the active mode.

### property voltage\_ac\_bandwidth

Control (floating) the AC voltage detector bandwidth in Hz, which can take the values 3, 30, and 300 Hz.

### property voltage\_ac\_digits

Control (integer) the number of digits in the AC voltage readings, which can take values from 4 to 7.

## property voltage\_ac\_nplc

Control (floating) the number of power line cycles (NPLC) for the AC voltage measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property voltage\_ac\_range

Control (floating) the AC voltage range in Volts, which can take values from 0 to 757.5 V. Auto-range is disabled when this property is set.

### property voltage\_ac\_reference

Control (floating) the AC voltage reference value in Volts, which can take values from -757.5 to 757.5 Volts.

## property voltage\_digits

Control (integer) the number of digits in the DC voltage readings, which can take values from 4 to 7.

## property voltage\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC voltage measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

## property voltage\_range

Control (floating) the DC voltage range in Volts, which can take values from 0 to 1010 V. Auto-range is disabled when this property is set.

### property voltage\_reference

Control (floating) the DC voltage reference value in Volts, which can take values from -1010 to 1010 V.

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

#### wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

#### **Parameters**

• **should\_stop** – A function that returns True when this function should return early

- **timeout** A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

## 7.31.2 Keithley 2200 Series Power Supplies

```
class pymeasure.instruments.keithley.Keithley2200(adapter, name='Keithley2200', **kwargs)
```

Bases: SCPIUnknownMixin, Instrument

Represents the Keithley 2200 Power Supply.

 $ch_1$ 

## Channel

**PSChannel** 

ch\_2

## Channel

**PSChannel** 

 $ch_3$ 

#### Channel

**PSChannel** 

## class BaseChannelCreator(cls, \*\*kwargs)

Bases: object

Base class for ChannelCreator and MultiChannelCreator.

#### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- \*\*kwargs Keyword arguments for all children.

### class ChannelCreator(cls, id=None, \*\*kwargs)

Bases: BaseChannelCreator

Add a single channel to the parent class.

The child will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name that ChannelCreator was assigned to in the *Instrument* class will be the name of the channel interface.

```
class Extreme5000(Instrument):
    # Two output channels, accessible by their property names
    # and both are accessible through the 'channels' collection
    output_A = Instrument.ChannelCreator(Extreme5000Channel, "A")
    output_B = Instrument.ChannelCreator(Extreme5000Channel, "B")
    # A channel without a channel accessible through the 'motor' collection
    motor = Instrument.ChannelCreator(MotorControl)

inst = SomeInstrument()
# Set the extreme_temp for channel A of Extreme5000 instrument
inst.output_A.extreme_temp = 42
```

#### **Parameters**

- cls Channel class for channel interface
- id The id of the channel on the instrument, integer or string.
- \*\*kwargs Keyword arguments for all children.

## class MultiChannelCreator(cls, id=None, prefix='ch\_', \*\*kwargs)

Bases: BaseChannelCreator

Add channels to the parent class.

The children will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name (e.g. channels) will be used as the *collection* of the children. You may define the attribute prefix. If there are no other pressing reasons, use channels as the attribute name and leave the prefix at the default "ch\_".

## **Parameters**

- cls Class for all children or tuple/list of classes, one for each child.
- id tuple/list of ids of the channels on the instrument.
- **prefix** Collection prefix for the attributes, e.g. "*ch*\_" creates attribute *self.ch*\_A. If prefix evaluates False, the child will be added directly under the variable name. Required if id is tuple/list.

• **\*\*kwargs** – Keyword arguments for all children.

add\_child(cls, id=None, collection='channels', prefix='ch', attr name='', \*\*kwargs)

Add a child to this instance and return its index in the children list.

The newly created child may be accessed either by the id in the children dictionary or by the created attribute, e.g. the fifth channel of instrument with id "F" has two access options: instrument.channels["F"] == instrument.ch\_F

#### 1 Note

Do not change the default *collection* or *prefix* parameter, unless you have to distinguish several collections of different children, e.g. different channel types (analog and digital).

#### **Parameters**

- **cls** Class of the channel.
- **id** Child id how it is used in communication, e.g. "A".
- collection Name of the collection of children, used for dictionary access to the channel interfaces.
- **prefix** For creating multiple channel interfaces, the prefix e.g. "ch\_" is prepended to the attribute name of the channel interface self.ch\_A. If prefix evaluates False, the child will be added directly under the collection name.
- attr\_name For creating a single channel interface, the attr\_name argument is used when setting the attribute name of the channel interface.
- **\*\*kwargs** Keyword arguments for the channel creator.

## **Returns**

Instance of the created child.

## binary\_values(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return a numpy array of the binary data.

#### **Parameters**

- **command** Command to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- **kwargs** Arguments for read\_binary\_values().

#### Returns

NumPy array of values.

## check\_errors()

Read all errors from the instrument.

## **Returns**

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

## check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### **Returns**

List of error entries.

## property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## property display\_enabled

Control whether the display is enabled.

## property display\_text\_data

Control text to be displayed(32 characters).

## static get\_channel\_pairs(cls)

Return a list of all the Instrument's channel pairs

#### static get\_channels(cls)

Return a list of all the Instrument's ChannelCreator and MultiChannelCreator instances

#### property id

Get the identification of the instrument.

### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

## property options

Get the device options installed.

## read\_binary\_values(\*\*kwargs)

Read binary values from the device.

## read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

## **Returns bytes**

Bytes response of the instrument (including termination).

## remove\_child(child)

Remove the child from the instrument and the corresponding collection.

### **Parameters**

child - Instance of the child to delete.

#### reset()

Reset the instrument.

#### shutdown()

Brings the instrument to a safe and stable state

## property status

Get the status byte and Master Summary Status bit.

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

## write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

## write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

## class pymeasure.instruments.keithley.keithley2200.PSChannel(parent, id)

Bases: Channel

Implementation of a Keithley 2200 channel.

## property current

Measure the current in Amps.

## property current\_limit

Control output current in Amps.

## insert\_id(command)

Insert the channel id in a command replacing *placeholder*.

Subclass this method if you want to do something else, like always prepending the channel id.

## property output\_enabled

Control the output state.

## property power

Measure the power in watts.

## property voltage

Measure the voltage in Volts.

## property voltage\_limit

Control the maximum voltage that can be set.

### property voltage\_limit\_enabled

Control whether the maximum voltage limit is enabled.

## property voltage\_setpoint

Control output voltage in Volts.

## 7.31.3 Keithley 2260B DC Power Supply

**class** pymeasure.instruments.keithley.**Keithley2260B**(adapter, name='Keithley 2260B DC Power Supply', read\_termination=\n', \*\*kwargs)

Bases: SCPIMixin, Instrument

Represents the Keithley 2260B Power Supply (minimal implementation) and provides a high-level interface for interacting with the instrument.

For a connection through tcpip, the device only accepts connections at port 2268, which cannot be configured otherwise. example connection string: 'TCPIP::xxx.xxx.xxx.:2268::SOCKET' the read termination for this interface is

```
source = Keithley2260B("GPIB::1")
source.voltage = 1
print(source.voltage)
print(source.current)
print(source.power)
print(source.applied)
```

### property applied

Control voltage (volts) and current (amps) simultaneously. Values need to be supplied as tuple of (voltage, current). Depending on whether the instrument is in constant current or constant voltage mode, the values achieved by the instrument will differ from the ones set.

#### check\_errors()

Read all errors from the instrument.

### Returns

List of error entries.

## check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

## clear()

Clear the instrument status byte.

## property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

#### property current

Get the current (in Ampere) the dc power supply is putting out.

## property current\_limit

Control the source current in amps. This is not checked against the allowed range. Depending on whether the instrument is in constant current or constant voltage mode, this might differ from the actual current achieved. (float)

### property enabled

Control whether the output is enabled, see *output\_enabled*.

#### property error

Get the next error of the instrument (list of code and message).

#### property id

Get the identification of the instrument.

### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

#### property options

Get the device options installed.

#### property output\_enabled

Control whether the source is enabled, takes values True or False. (bool)

### property power

Get the power (in Watt) the dc power supply is putting out.

### read(\*\*kwargs)

Read up to (excluding) *read\_termination* or the whole read buffer.

## read\_binary\_values(\*\*kwargs)

Read binary values from the device.

#### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

## reset()

Reset the instrument.

#### shutdown()

Disable output, call parent function

#### property status

Get the status byte and Master Summary Status bit.

### property voltage

Get the voltage (in Volt) the dc power supply is putting out.

### property voltage\_setpoint

Control the source voltage in volts. This is not checked against the allowed range. Depending on whether the instrument is in constant current or constant voltage mode, this might differ from the actual voltage achieved. (float)

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### Parameters

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

## 7.31.4 Keithley 2281S power supply and battery simulator / characterizer

```
class pymeasure.instruments.keithley.Keithley2281S(adapter, name='Keithley2281S', **kwargs)
```

```
Bases: SCPIMixin, Instrument, KeithleyBuffer
```

Represents the Keithley 2281S-20-6 power supply and battery simulator / characterizer. Common commands beside *function\_mode* and power supply commands should also work for Keithley 2280S power supplies, although this is untested.

bs

## Channel

BatterySimulationChannel

bt

#### Channel

BatteryTestChannel

ps

## Channel

PowerSupplyChannel

## property buffer\_data

Get a pandas dataframe of values from the buffer.

### property buffer\_points

Control the maximum number of buffer points to store.

**characterize**(lower\_voltage: float, upper\_voltage: float, charge\_current: float, memory\_slot: int, model\_voltage\_offset: float = 0.05, charge\_delay: float = 0)

Convenience function for testing a battery and saving its model to the internal memory.

The device can only discharge the battery at a fixed 1A! If this current is too high, a series resistor has to be used during discharge to limit the current! If the battery is discharged below the lower limit, it will be charged with a 10th of the set charge current till it reaches the lower limit, then the battery profile will be characterized. The function will block until the end of the measurement!

#### **Parameters**

- **lower\_voltage** (*float*) discharge end voltage, set this slightly lower (~0.05V) than in normal operation
- **upper\_voltage** (*float*) charge end voltage, set this slightly higher (~0.05V) than in normal operation
- **charge\_current** (*float*) maximum charge current. A 1/100th is used as (dis-)charge end current.
- **memory\_slot** (*int*) Internal memory slot to save the model to
- model\_voltage\_offset (float, optional) Voltage offset for the generated model upper and lower limits. These have to be in the range of the actual measured values, i.e. in between the hard limits of the measurement. Defaults to 0.05
- **charge\_delay**(*float*, *optional*)—Seconds to wait between discharging and charging. Actual time is higher since the device has some internal delay on reporting end of a test. Defaults to 0.

### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### clear()

Clear the instrument status byte.

#### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## property display\_text\_data

Set control text to be displayed(24 characters).

## property function\_mode

Control function mode to use.

Valid values are "POWER", "TEST" and "SIMULATOR".

## property id

Get the identification of the instrument.

## property line\_frequency

Get line frequency.

## property measurement\_event

Get measurement event register.

## property measurement\_ongoing: bool

Get measurement status.

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

## property options

Get the device options installed.

## read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

## read\_binary\_values(\*\*kwargs)

Read binary values from the device.

## read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

## **Returns bytes**

Bytes response of the instrument (including termination).

## property reading\_available: bool

Get availability of a reading.

## reset()

Reset the instrument.

## shutdown()

Brings the instrument to a safe and stable state

#### property status

Get the status byte and Master Summary Status bit.

## property summary\_event

Get summary event register.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

 $\textbf{class} \ \ \textbf{pymeasure.instruments.keithley.keithley2281S.} \textbf{\textit{Keithley2281SOperationEventRegister}} (\textit{value}, \textit{value}, \textit{value},$ 

```
names=<not
given>,
*val-
ues,
mod-
ule=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

Enum containing Keithley 2281S Operation Instrument Summary Event Register definition

class pymeasure.instruments.keithley.keithley2281S.Keithley2281SMeasurementEventRegister(value,

names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Enum containing Keithley 2281S Measurement Instrument Summary Event Register definition

## 7.31.5 Keithley 2306 Dual Channel Battery/Charger Simulator

class pymeasure.instruments.keithley.Keithley2306(adapter, name='Keithley 2306', \*\*kwargs)

Bases: SCPIUnknownMixin, Instrument

Represents the Keithley 2306 Dual Channel Battery/Charger Simulator.

## property both\_channels\_enabled

Set whether both channel outputs are enabled, takes values of True or False.

ch(channel\_number)

Get a channel from this instrument.

## **Param**

channel\_number: int: the number of the channel to be selected

## **Type**

Keithley2306Channel

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### **Returns**

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

## Returns

List of error entries.

#### clear()

Clear the instrument status byte.

### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## property display\_brightness

Control (floating) the display brightness, takes values between 0.0 and 1.0. A blank display is 0.0, 1/4 brightness is for values less or equal to 0.25, otherwise 1/2 brightness for values less than or equal to 0.5, otherwise 3/4 brightness for values less than or equal to 0.75, otherwise full brightness.

## property display\_channel

Control (integer) the display channel, takes values 1 or 2.

## property display\_enabled

Control (boolean) whether the display is enabled, takes values True or False.

## property display\_text\_data

Control text to be displayed, takes strings up to 32 characters.

## property display\_text\_enabled

Control (boolean) whether display text is enabled, takes values True or False.

### property id

Get the identification of the instrument.

## property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

## property options

Get the device options installed.

## read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

## read\_binary\_values(\*\*kwargs)

Read binary values from the device.

## read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

## **Returns bytes**

Bytes response of the instrument (including termination).

### relay(relay\_number)

Get a relay channel from this instrument.

#### **Param**

relay number: int: the number of the relay to be selected

```
Type
Relay
```

#### reset()

Reset the instrument.

#### shutdown()

Brings the instrument to a safe and stable state

## property status

Get the status byte and Master Summary Status bit.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- values The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

## 7.31.6 Keithley 2400 SourceMeter

Bases: KeithleyBuffer, SCPIMixin, Instrument

Represent the Keithley 2400 SourceMeter and provide a high-level interface for interacting with the instrument.

```
keithley = Keithley2400("GPIB::1")

keithley.apply_current()  # Sets up to source current
keithley.source_current_range = 10e-3  # Sets the source current range to 10 mA
keithley.compliance_voltage = 10  # Sets the compliance voltage to 10 V
keithley.source_current = 0  # Sets the source current to 0 mA
keithley.enable_source()  # Enables the source output

keithley.measure_voltage()  # Sets up to measure voltage
```

(continues on next page)

(continued from previous page)

```
keithley.ramp_to_current(5e-3)  # Ramps the current to 5 mA
print(keithley.voltage)  # Prints the voltage in Volts

keithley.shutdown()  # Ramps the current to 0 mA and disables_
output
```

### apply\_current(current range=None, compliance voltage=0.1)

Configure the instrument to apply a source current, and uses an auto range unless a current range is specified. The compliance voltage is also set.

#### **Parameters**

- compliance\_voltage A float in the correct range for a compliance\_voltage
- current\_range A current\_range value or None

## apply\_voltage(voltage\_range=None, compliance\_current=0.1)

Configure the instrument to apply a source voltage, and uses an auto range unless a voltage range is specified. The compliance current is also set.

#### **Parameters**

- compliance\_current A float in the correct range for a compliance\_current
- voltage\_range A voltage\_range value or None

## property auto\_output\_off

Control whether auto output-off is activated. Valid values are True (output off after measurement) and False (output stays on after measurement).

### auto\_range\_source()

Configure the source to use an automatic range.

#### property auto\_zero

Control whether the auto zero option is enabled. Valid values are True (enabled) and False (disabled) and 'ONCE' (force immediate).

## beep(frequency, duration)

Sound a system beep.

#### **Parameters**

- **frequency** A frequency in Hz between 65 Hz and 2 MHz
- duration A time in seconds between 0 and 7.9 seconds

## property buffer\_data

Get a numpy array of values from the buffer.

## property buffer\_points

Control (integer) the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

## check\_errors()

Read all errors from the instrument.

#### **Returns**

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### **Returns**

List of error entries.

## clear()

Clear the instrument status byte.

## property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## property compliance\_current

Control (floating) the compliance current in Amps.

## property compliance\_voltage

Control the compliance voltage in Volts (float, truncated from -210 to 210).

### config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

#### **Parameters**

- **points** The number of points in the buffer.
- **delay** The delay time in seconds.

#### property current

Get the current in Amps if configured (float).

#### property current\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC current measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

## property current\_range

Control the measurement current range in Amps (float, truncated from -1.05 to 1.05). Auto-range is disabled when this property is set.

## disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

### disable\_output\_trigger()

Disable the output trigger for the Trigger layer

## disable\_source()

Disable the source of current or voltage depending on the configuration of the instrument.

## property display\_enabled

Control whether or not the display of the sourcemeter is enabled. Valid values are True and False.

#### enable\_source()

Enable the source of current or voltage depending on the configuration of the instrument.

#### property error

Get the next error from the queue.

Deprecated since version 0.15: Use next\_error instead.

## property filter\_count

Control the number of readings that are acquired and stored in the filter buffer (integer, truncated from 1 to 100).

## property filter\_state

Control if the filter is active (string, strictly 'ON' or 'OFF').

## property filter\_type

Control (String) the filter's type. REP: Repeating filter MOV: Moving filter

#### property id

Get the identification of the instrument.

## is\_buffer\_full()

Return True if the buffer is full of measurements.

#### property line\_frequency

Control the line frequency in Hertz (integer, strictly 50 or 60).

## property line\_frequency\_auto

Control the auto line frequency (boolean).

#### property max\_current

Get the maximum current from the buffer.

## property max\_resistance

Get the maximum resistance from the buffer.

#### property max\_voltage

Get the maximum voltage from the buffer.

### property maximums

Get the calculated maximums for voltage, current, and resistance from the buffer data as a list (list of floats).

#### property mean\_current

Get the mean current from the buffer.

### property mean\_resistance

Get the mean resistance from the buffer.

### property mean\_voltage

Get the mean voltage from the buffer.

### property means

Get the calculated means for voltage, current, and resistance from the buffer data as a list (list of floats).

## property measure\_concurent\_functions

Control the ability to measure more than one function simultaneously (boolean).

measure\_current(nplc=1, current=0.000105, auto\_range=True)

Configure the measurement of current.

### **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- current Upper limit of current in Amps, from -1.05 A to 1.05 A
- auto\_range Enables auto\_range if True, else uses the set current

measure\_resistance(nplc=1, resistance=210000.0, auto\_range=True)

Configure the measurement of resistance.

#### **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- resistance Upper limit of resistance in Ohms, from -210 MOhms to 210 MOhms
- auto\_range Enables auto\_range if True, else uses the set resistance

measure\_voltage(nplc=1, voltage=21.0, auto\_range=True)

Configure the measurement of voltage.

#### **Parameters**

- **nplc** Number of power line cycles (NPLC) from 0.01 to 10
- voltage Upper limit of voltage in Volts, from -210 V to 210 V
- auto\_range Enables auto\_range if True, else uses the set voltage

#### property min\_current

Get the minimum current from the buffer.

## property min\_resistance

Get the minimum resistance from the buffer.

### property min\_voltage

Get the minimum voltage from the buffer.

#### property minimums

Get the calculated minimums for voltage, current, and resistance from the buffer data as a list (list of floats).

## property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

## property options

Get the device options installed.

## property output\_off\_state

Control the output-off state of the SourceMeter. HIMP: output relay is open, disconnects external circuitry. NORM: V-Source is selected and set to 0V, Compliance is set to 0.5% full scale of the present current range. ZERO: V-Source is selected and set to 0V, compliance is set to the programmed Source I value or to 0.5% full scale of the present current range, whichever is greater. GUAR: I-Source is selected and set to 0A

### output\_trigger\_on\_external(line=1, after='DEL')

Configure the output trigger on the specified trigger link line number, with the option of supplying the part of the measurement after which the trigger should be generated (default to delay, which is right before the measurement)

#### **Parameters**

- line A trigger line from 1 to 4
- **after** An event string that determines when to trigger

## ramp\_to\_current(target\_current, steps=30, pause=0.02)

Ramp to a target current from the set current value over a certain number of linear steps, each separated by a pause duration.

#### **Parameters**

- target\_current A current in Amps
- **steps** An integer number of steps
- pause A pause duration in seconds to wait between steps

### ramp\_to\_voltage(target voltage, steps=30, pause=0.02)

Ramp to a target voltage from the set voltage value over a certain number of linear steps, each separated by a pause duration.

#### **Parameters**

- target\_voltage A voltage in Amps
- **steps** An integer number of steps
- pause A pause duration in seconds to wait between steps

## read(\*\*kwargs)

Read up to (excluding) *read\_termination* or the whole read buffer.

## read\_binary\_values(\*\*kwargs)

Read binary values from the device.

## read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

## **Returns bytes**

Bytes response of the instrument (including termination).

### reset()

Reset the instrument and clear the queue.

## reset\_buffer()

Reset the buffer.

## property resistance

Get the resistance in Ohms, if configured for this reading.

### property resistance\_nplc

Control the number of power line cycles (NPLC) for the 2-wire resistance measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

#### property resistance\_range

Control the resistance range in Ohms (float, truncated from 0 to 210e6). Auto-range is disabled when this property is set.

## sample\_continuously()

Cause the instrument to continuously read samples and turns off any buffer or output triggering

#### set\_timed\_arm(interval)

Set up the measurement to be taken with the internal trigger at a variable sampling rate defined by the interval in seconds between sampling points

## set\_trigger\_counts(arm, trigger)

Set the number of counts for both the sweeps (arm) and the points in those sweeps (trigger), where the total number of points can not exceed 2500

### shutdown()

Ensure that the current or voltage is turned to zero and disable the output.

#### property source\_current

Control the source current in Amps (float, truncated from -1.05 to 1.05).

### property source\_current\_range

Control the source current range in Amps (float, truncated from -1.05 to 1.05). Auto-range is disabled when this property is set.

## property source\_delay

Control (floating) a manual delay for the source after the output is turned on before a measurement is taken. When this property is set, the auto delay is turned off. Valid values are between 0 [seconds] and 999.9999 [seconds].

#### property source\_delay\_auto

Control the auto delay (boolean).

## property source\_enabled

Control whether the source is enabled, takes values True or False. The convenience methods <code>enable\_source()</code> and <code>disable\_source()</code> can also be used.

## property source\_mode

Control (string) the source mode, which can take the values 'current' or 'voltage'. The convenience methods apply\_current() and apply\_voltage() can also be used.

## property source\_voltage

Control the source voltage in Volts (float).

### property source\_voltage\_range

Control the source voltage range in Volts (float, truncated from -210 to 210). Auto-range is disabled when this property is set.

## property standard\_devs

Get the calculated standard deviations for voltage, current, and resistance from the buffer data as a list (list of floats).

#### start\_buffer()

Starts the buffer.

#### status()

Get the status byte and Master Summary Status bit.

### property std\_current

Get the current standard deviation from the buffer.

#### property std\_resistance

Get the resistance standard deviation from the buffer.

### property std\_voltage

Get the voltage standard deviation from the buffer.

## stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

#### **triad**(base frequency, duration)

Sound a musical triad using the system beep.

#### **Parameters**

- base\_frequency A frequency in Hz between 65 Hz and 1.3 MHz
- **duration** A time in seconds between 0 and 7.9 seconds

#### trigger()

Execute a bus trigger, which can be used when trigger\_on\_bus() is configured.

## property trigger\_count

Control the trigger count (integer, truncated from 1 to 2500).

## property trigger\_delay

Control the trigger delay in seconds (float, truncated from 0 to 999.9999).

## trigger\_immediately()

Configure measurements to be taken with the internal trigger at the maximum sampling rate.

## trigger\_on\_bus()

Configure the trigger to detect events based on the bus trigger, which can be activated by trigger().

## trigger\_on\_external(line=1)

Configure the measurement trigger to be taken from a specific line of an external trigger

### **Parameters**

**line** – A trigger line from 1 to 4

### use\_front\_terminals()

Enable the front terminals for measurement, and disable the rear terminals.

## use\_rear\_terminals()

Enable the rear terminals for measurement, and disable the front terminals.

### property voltage

Get the voltage in Volts if configured (float).

## property voltage\_nplc

Control the number of power line cycles (NPLC) for the DC voltage measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

## property voltage\_range

Control the measurement voltage range in Volts (float, truncated from -210 to 210). Auto-range is disabled when this property is set.

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

#### **Parameters**

- **should\_stop** A function that returns True when this function should return early
- timeout A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

## property wires

Control the number of wires in use for resistance measurements (integer, strictly 2 or 4).

write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

write\_bytes(content, \*\*kwargs)

Write the bytes content to the instrument.

## 7.31.7 Keithley 2450 SourceMeter

Bases: KeithleyBuffer, SCPIMixin, Instrument

Represents the Keithley 2450 SourceMeter and provides a high-level interface for interacting with the instrument.

NOTE: The default buffer handling SCPI command set of the Keithley 2450 model is currently unsupported. The instrument works if made to emulate a 2400 model by setting its command set to "SCPI 2400" through the instrument's system settings.

```
keithley = Keithley2450("GPIB::1")
keithley.apply_current()
                                       # Sets up to source current
keithley.source_current_range = 10e-3  # Sets the source current range to 10 mA
keithley.compliance_voltage = 10  # Sets the compliance voltage to 10 V
keithley.source_current = 0
                                      # Sets the source current to 0 mA
keithley.enable_source()
                                      # Enables the source output
keithley.measure_voltage()
                                       # Sets up to measure voltage
keithley.ramp_to_current(5e-3)
                                       # Ramps the current to 5 mA
print(keithley.voltage)
                                       # Prints the voltage in Volts
                                       # Ramps the current to 0 mA and disables.
keithley.shutdown()
→output
```

## apply\_current(current\_range=None, compliance\_voltage=0.1)

Configures the instrument to apply a source current, and uses an auto range unless a current range is specified. The compliance voltage is also set.

### **Parameters**

- compliance\_voltage A float in the correct range for a compliance\_voltage
- current\_range A current\_range value or None

```
apply_voltage(voltage_range=None, compliance_current=0.1)
```

Configures the instrument to apply a source voltage, and uses an auto range unless a voltage range is specified. The compliance current is also set.

#### **Parameters**

- compliance\_current A float in the correct range for a compliance\_current
- voltage\_range A voltage\_range value or None

## auto\_range\_source()

Configures the source to use an automatic range.

beep(frequency, duration)

Sounds a system beep.

#### **Parameters**

- **frequency** A frequency in Hz between 65 Hz and 2 MHz
- **duration** A time in seconds between 0 and 7.9 seconds

### property buffer\_data

Get a numpy array of values from the buffer.

## property buffer\_points

Control (integer) the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

## check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### **Returns**

List of error entries.

## clear()

Clear the instrument status byte.

## property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

## property compliance\_current

Control (floating) the compliance current in Amps.

#### property compliance\_voltage

Control (floating) the compliance voltage in Volts.

## config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

### **Parameters**

- **points** The number of points in the buffer.
- **delay** The delay time in seconds.

## property current

Get the current in Amps, if configured for this reading.

#### property current\_filter\_count

Control (integer) the number of readings that are acquired and stored in the filter buffer for the averaging

### property current\_filter\_state

Control (string) if the filter is active.

### property current\_filter\_type

Control (String) the filter's type for the current. REP: Repeating filter MOV: Moving filter

### property current\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC current measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property current\_output\_off\_state

Control SourceMeter output. HIMP: output relay is open, disconnects external circuitry. NORM: V-Source is selected and set to 0V, Compliance is set to 0.5% full scale of the present current range. ZERO: V-Source is selected and set to 0V, compliance is set to the programmed Source I value or to 0.5% full scale of the present current range, whichever is greater. GUAR: I-Source is selected and set to 0A

#### property current\_range

Control (floating) the measurement current range in Amps, which can take values between -1.05 and +1.05 A. Auto-range is disabled when this property is set.

### disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

#### disable\_source()

Disables the source of current or voltage depending on the configuration of the instrument.

## enable\_source()

Enables the source of current or voltage depending on the configuration of the instrument.

### property error

Get the next error from the queue.

Deprecated since version 0.15: Use *next\_error* instead.

## property id

Get the identification of the instrument.

#### is\_buffer\_full()

Return True if the buffer is full of measurements.

### property max\_current

Get the maximum current from the buffer

## property max\_resistance

Get the maximum resistance from the buffer

#### property max\_voltage

Get the maximum voltage from the buffer

### property maximums

Get the calculated maximums for voltage, current, and resistance from the buffer data as a list.

## property mean\_current

Get the mean current from the buffer

# property mean\_resistance

Get the mean resistance from the buffer

# property mean\_voltage

Get the mean voltage from the buffer

### property means

Get the calculated means (averages) for voltage, current, and resistance from the buffer data as a list.

measure\_current(nplc=1, current=0.000105, auto range=True)

Configures the measurement of current.

# **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- current Upper limit of current in Amps, from -1.05 A to 1.05 A
- auto\_range Enables auto\_range if True, else uses the set current

measure\_resistance(nplc=1, resistance=210000.0, auto range=True)

Configures the measurement of resistance.

#### **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- resistance Upper limit of resistance in Ohms, from -210 MOhms to 210 MOhms
- auto\_range Enables auto\_range if True, else uses the set resistance

measure\_voltage(nplc=1, voltage=21.0, auto\_range=True)

Configures the measurement of voltage.

# **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- voltage Upper limit of voltage in Volts, from -210 V to 210 V
- auto\_range Enables auto\_range if True, else uses the set voltage

# property min\_current

Get the minimum current from the buffer

### property min\_resistance

Get the minimum resistance from the buffer

### property min\_voltage

Get the minimum voltage from the buffer

# property minimums

Get the calculated minimums for voltage, current, and resistance from the buffer data as a list.

### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# property options

Get the device options installed.

### ramp\_to\_current(target\_current, steps=30, pause=0.02)

Ramps to a target current from the set current value over a certain number of linear steps, each separated by a pause duration.

#### **Parameters**

- target\_current A current in Amps
- **steps** An integer number of steps
- pause A pause duration in seconds to wait between steps

### ramp\_to\_voltage(target\_voltage, steps=30, pause=0.02)

Ramps to a target voltage from the set voltage value over a certain number of linear steps, each separated by a pause duration.

#### **Parameters**

- target\_voltage A voltage in Amps
- **steps** An integer number of steps
- pause A pause duration in seconds to wait between steps

### read(\*\*kwargs)

Read up to (excluding) *read\_termination* or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

# reset()

Resets the instrument and clears the queue.

# reset\_buffer()

Reset the buffer.

# property resistance

Get the resistance in Ohms, if configured for this reading.

# property resistance\_nplc

Control (floating) the number of power line cycles (NPLC) for the 2-wire resistance measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property resistance\_range

Control (floating) the resistance range in Ohms, which can take values from 0 to 210 MOhms. Auto-range is disabled when this property is set.

### shutdown()

Ensures that the current or voltage is turned to zero and disables the output.

### property source\_current

Control (floating) the source current in Amps.

#### property source\_current\_delay

Control (floating) a manual delay for the source after the output is turned on before a measurement is taken. When this property is set, the auto delay is turned off. Valid values are between 0 [seconds] and 999.9999 [seconds].

### property source\_current\_delay\_auto

Control (bool) auto delay. Valid values are True and False.

### property source\_current\_range

Control (floating) the source current range in Amps, which can take values between -1.05 and +1.05 A. Auto-range is disabled when this property is set.

### property source\_enabled

Get a boolean value that is True if the source is enabled.

# property source\_mode

Control (string) the source mode, which can take the values 'current' or 'voltage'. The convenience methods apply\_current() and apply\_voltage() can also be used.

# property source\_voltage

Control (floating) the source voltage in Volts.

# property source\_voltage\_delay

Control (floating) a manual delay for the source after the output is turned on before a measurement is taken. When this property is set, the auto delay is turned off. Valid values are between 0 [seconds] and 999.9999 [seconds].

### property source\_voltage\_delay\_auto

Control (bool) auto delay. Valid values are True and False.

#### property source\_voltage\_range

Control (floating) the source voltage range in Volts, which can take values from -210 to 210 V. Auto-range is disabled when this property is set.

### property standard\_devs

Get the calculated standard deviations for voltage, current, and resistance from the buffer data as a list.

### start\_buffer()

Starts the buffer.

# property status

Get the status byte and Master Summary Status bit.

### property std\_current

Get the current standard deviation from the buffer

### property std\_resistance

Get the resistance standard deviation from the buffer

# property std\_voltage

Get the voltage standard deviation from the buffer

### stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

### triad(base\_frequency, duration)

Sounds a musical triad using the system beep.

#### **Parameters**

- base\_frequency A frequency in Hz between 65 Hz and 1.3 MHz
- duration A time in seconds between 0 and 7.9 seconds

### trigger()

Executes a bus trigger.

# use\_front\_terminals()

Enables the front terminals for measurement, and disables the rear terminals.

# use\_rear\_terminals()

Enables the rear terminals for measurement, and disables the front terminals.

### property voltage

Get the voltage in Volts, if configured for this reading.

# property voltage\_filter\_count

Control (integer) the number of readings that are acquired and stored in the filter buffer for the averaging

# property voltage\_filter\_type

Control (String) the filter's type for the current. REP: Repeating filter MOV: Moving filter

# property voltage\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC voltage measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property voltage\_output\_off\_state

Control the SourceMeter output. HIMP: output relay is open, disconnects external circuitry. NORM: V-Source is selected and set to 0V, Compliance is set to 0.5% full scale of the present current range. ZERO: V-Source is selected and set to 0V, compliance is set to the programmed Source I value or to 0.5% full scale of the present current range, whichever is greater. GUAR: I-Source is selected and set to 0A

### property voltage\_range

Control (floating) the measurement voltage range in Volts, which can take values from -210 to 210 V. Auto-range is disabled when this property is set.

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

### wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

### **Parameters**

• **should\_stop** – A function that returns True when this function should return early

- timeout A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

# property wires

Control (integer) the number of wires in use for resistance measurements, which can take the value of 2 or 4.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.31.8 Keithley 2510 TEC SourceMeter

Bases: SCPIMixin, Instrument

Represents the Keithley 2510 TEC Sourcemeter and provides a high-level interface for interacting with the instrument.

# check\_errors()

Read all errors from the instrument.

### Returns

List of error entries.

# check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

#### check set errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

# check\_temperature\_stability(tolerance=0.1, period=10, points=64)

Determine whether the temperature is stable at the temperature setpoint over a specified period.

#### **Parameters**

- **tolerance** Maximum allowed deviation from temperature setpoint, in degrees Centigrade.
- **period** Time period over which stability is checked, in seconds.

#### Returns

True if stable, False otherwise.

### clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property current

Measure the DC current through the thermoelectric cooler, in Amps.

### disable\_source()

Disables the source.

# disable\_temperature\_protection()

Disable temperature protection.

# enable\_source()

Enables the source.

### enable\_temperature\_protection()

Enable temperature protection.

# property id

Get the identification of the instrument.

### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

### property options

Get the device options installed.

# read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument.

# shutdown()

Brings the instrument to a safe and stable state

# property source\_enabled

Control whether the source is enabled (bool). The convenience methods <code>enable\_source()</code> and <code>disable\_source()</code> can also be used.

# property status

Get the status byte and Master Summary Status bit.

### property temperature

Measure the temperature using the thermistor, in degrees centigrade.

### property temperature\_pid

Control the temperature PID loop constants through the tuple (proportional\_const, integral\_const, derivative\_const).

# property temperature\_pid\_d

Control the derivative constant of the temperature PID loop.

# property temperature\_pid\_i

Control the integral constant of the temperature PID loop.

# property temperature\_pid\_p

Control the proportional constant of the temperature PID loop.

### property temperature\_protection\_enabled

Control whether temperature protection is enabled. Takes values True or False. The convenience methods <code>enable\_temperature\_protection()</code> and <code>disable\_temperature\_protection()</code> can also be used.

# property temperature\_protection\_high

Control the upper temperature limit in degrees centigrade (float strictly from -50 to 225)

### property temperature\_protection\_low

Control the lower temperature limit in degrees centigrade (float strictly from -50 to 225)

# property temperature\_protection\_range

Control the lower and upper temperature limits in degrees centigrade through the tuple (lower\_limit, upper\_limit).

### property temperature\_setpoint

Control the temperature setpoint, in degrees centigrade (float strictly from -50 to 225)

### property voltage

Measure the DC voltage through the thermoelectric cooler, in Volts.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

Block the program, waiting for the temperature to stabilize at the temperature setpoint.

#### **Parameters**

- **tolerance** Maximum allowed deviation from temperature setpoint, in degrees Centigrade.
- **period** Time period over which stability is checked, in seconds.
- **should\_stop** Function that returns True to stop waiting.
- **timeout** Maximum waiting time, in seconds.

#### Returns

True when stable, False if stopped by should\_stop.

### Raises

**TimeoutError** – If the temperature does not stabilize within the timeout period.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.31.9 Keithley 2600 SourceMeter

Bases: SCPIUnknownMixin, Instrument

Represents the Keithley 2600 series (channel A and B) SourceMeter

#### check errors()

Read all errors from the instrument.

### Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property error

Get the next error from the queue.

Deprecated since version 0.15: Use *next\_error* instead.

### property id

Get the identification of the instrument.

# property next\_error

Get a tuple of an error code and message from a single error.

### property options

Get the device options installed.

```
read(**kwargs)
```

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

# **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- kwargs Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument.

### shutdown()

Brings the instrument to a safe and stable state

# property status

Get the status byte and Master Summary Status bit.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.31.10 Keithley 2700 MultiMeter/Switch System

Bases: KeithleyBuffer, SCPIMixin, Instrument

Represents the Keithley 2700 Multimeter/Switch System and provides a high-level interface for interacting with the instrument.

```
keithley = Keithley2700("GPIB::1")
```

beep(frequency, duration)

Sounds a system beep.

### **Parameters**

- **frequency** A frequency in Hz between 65 Hz and 2 MHz
- **duration** A time in seconds between 0 and 7.9 seconds

### property buffer\_data

Get a numpy array of values from the buffer.

# property buffer\_points

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

# channels\_from\_rows\_columns(rows, columns, slot=None)

Determine the channel numbers between column(s) and row(s) of the 7709 connection matrix. Returns a list of channel numbers. Only one of the parameters 'rows' or 'columns' can be "all"

#### **Parameters**

- rows row number or list of numbers; can also be "all"
- columns column number or list of numbers; can also be "all"
- **slot** slot number (1 or 2) of the 7709 card to be used

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

# check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### clear()

Clear the instrument status byte.

# close\_rows\_to\_columns(rows, columns, slot=None)

Closes (connects) the channels between column(s) and row(s) of the 7709 connection matrix. Only one of the parameters 'rows' or 'columns' can be "all"

#### **Parameters**

- rows row number or list of numbers; can also be "all"
- columns column number or list of numbers; can also be "all"
- slot slot number (1 or 2) of the 7709 card to be used

# property closed\_channels

Control the opened and closed channels. All mentioned channels are closed, other channels will be opened.

### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

### **Parameters**

- **points** The number of points in the buffer.
- **delay** The delay time in seconds.

### determine\_valid\_channels()

Determine what cards are installed into the Keithley 2700 and from that determine what channels are valid.

### disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

### display\_closed\_channels()

Show the presently closed channels on the display of the Keithley 2700.

### property display\_text

Control (string) the text shown on the display of the Keithley 2700. Text can be up to 12 ASCII characters and must be enabled to show.

### property error

Get the next error from the queue.

Deprecated since version 0.15: Use *next\_error* instead.

# get\_state\_of\_channels(channels)

Get the open or closed state of the specified channels

### **Parameters**

**channels** – a list of channel numbers, or single channel number

# property id

Get the identification of the instrument.

### is\_buffer\_full()

Return True if the buffer is full of measurements.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# open\_all\_channels()

Open all channels of the Keithley 2700.

### property open\_channels

Set the specified list of channels. Can only be set.

### open\_rows\_to\_columns(rows, columns, slot=None)

Opens (disconnects) the channels between column(s) and row(s) of the 7709 connection matrix. Only one of the parameters 'rows' or 'columns' can be "all"

#### **Parameters**

- rows row number or list of numbers; can also be "all"
- columns column number or list of numbers; can also be "all"
- slot slot number (1 or 2) of the 7709 card to be used

# property options

Get the lists of the installed cards in the Keithley 2700. Returns a dict with the integer card numbers on the position.

# read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

### reset()

Resets the instrument and clears the queue.

# reset\_buffer()

Reset the buffer.

### shutdown()

Brings the instrument to a safe and stable state

### start\_buffer()

Starts the buffer.

# property status

Get the status byte and Master Summary Status bit.

### stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

# property text\_enabled

Control (boolean) whether a text message can be shown on the display of the Keithley 2700.

# triad(base\_frequency, duration)

Sounds a musical triad using the system beep.

#### **Parameters**

- base\_frequency A frequency in Hz between 65 Hz and 1.3 MHz
- duration A time in seconds between 0 and 7.9 seconds

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

#### **Parameters**

- **should\_stop** A function that returns True when this function should return early
- timeout A time in seconds after which this function should return early
- **interval** A time in seconds for how often to check if the buffer is full

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\**args*,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.31.11 Keithley 2750 Multimeter/Switch System

Bases: SCPIMixin, Instrument

Represents the Keithley2750 multimeter/switch system and provides a high-level interface for interacting with the instrument.

# check\_errors()

Read all errors from the instrument.

### Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

#### clear()

Clear the instrument status byte.

#### close(channel)

Closes (connects) the specified channel.

### **Parameters**

**channel** (int) – 3-digit number for the channel

### **Returns**

None

# property closed\_channels

Get the list of closed channels.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property id

Get the identification of the instrument.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# open(channel)

Opens (disconnects) the specified channel.

### **Parameters**

**channel** (int) – 3-digit number for the channel

# Returns

None

# open\_all()

Opens (disconnects) all the channels on the switch matrix.

### Returns

None

# property options

Get the device options installed.

### read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument.

### shutdown()

Brings the instrument to a safe and stable state

# property status

Get the status byte and Master Summary Status bit.

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

# write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

# **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

# write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

# 7.31.12 Keithley 6221 AC and DC Current Source

Bases: KeithleyBuffer, SCPIMixin, Instrument

Represents the Keithley 6221 AC and DC current source and provides a high-level interface for interacting with the instrument.

```
keithley = Keithley6221("GPIB::1")
keithlev.clear()
# Use the keithley as an AC source
keithley.waveform_function = "square"  # Set a square waveform
keithley.waveform_amplitude = 0.05  # Set the amplitude in Amps
keithley.waveform_offset = 0  # Set zero offset
keithley.source_compliance = 10  # Set compliance (limit) in V
keithley.waveform_dutycycle = 50  # Set duty cycle of wave in %
keithley.waveform_frequency = 347  # Set the frequency in Hz
keithley.waveform_ranging = "best"  # Set optimal output ranging
keithley.waveform_duration_cycles = 100 # Set duration of the waveform
# Link end of waveform to Service Request status bit
keithley.operation_event_enabled = 128 # OSB listens to end of wave
keithley.srq_event_enabled = 128  # SRQ listens to OSB
keithley.waveform_arm()
                                                      # Arm (load) the waveform
                                                       # Start the waveform
keithley.waveform_start()
keithley.adapter.wait_for_srq()
                                                       # Wait for the pulse to finish
                                                        # Disarm (unload) the waveform
keithley.waveform_abort()
keithley.shutdown()
                                                        # Disables output
```

**beep**(*frequency*, *duration*)

Sounds a system beep.

# **Parameters**

- **frequency** A frequency in Hz between 65 Hz and 2 MHz
- **duration** A time in seconds between 0 and 7.9 seconds

### property buffer\_data

Get a numpy array of values from the buffer.

# property buffer\_points

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

# check\_errors()

Read all errors from the instrument.

# Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

# **Parameters**

- points The number of points in the buffer.
- **delay** The delay time in seconds.

# define\_arbitary\_waveform(datapoints, location=1)

Define the data points for the arbitrary waveform and copy the defined waveform into the given storage location.

# **Parameters**

- **datapoints** a list (or numpy array) of the data points; all values have to be between -1 and 1; 100 points maximum.
- **location** integer storage location to store the waveform in. Value must be in range 1 to 4.

# delta\_abort()

Stop delta and place the Model 2182A in the local mode.

### delta\_arm()

Arm delta.

# property delta\_buffer\_points

Control the size of the buffer (integer strictly from 1 to 1000000).

Buffer size should be the same value as Delta count.

### property delta\_cold\_switch\_enabled

Control if cold switching mode is enabled (boolean).

# property delta\_compliance\_abort\_enabled

Control if compliance abort is enabled (boolean).

# property delta\_connected

Get connection status to 2182A.

### property delta\_cycles

Control the number of cycles to run for the delta measurements (integer strictly from 1 to 65536, or "INF").

# property delta\_delay

Control the delta delay in seconds (float strictly from 0 to 9999.999, or "INF").

# property delta\_high\_source

Control the delta high source value in A (float strictly from 0 to 0.105).

Set high source value will automatically set the low source value to minus the high source value.

# property delta\_low\_source

Control the delta low source value in A (float strictly from -0.105 to 0).

Usually no need to manually set this. By default, the low source value is minus the high source value.

# property delta\_measurement\_sets

Control the number of measurement sets to repeat for delta measurements (integer strictly from 1 to 65536, or "INF").

# property delta\_sense

Get the latest delta reading results from 2182/2182A.

### delta\_start()

Start delta measurements.

# property delta\_unit

Control the reading unit (string strictly in 'V', 'Ohms', 'W' and 'Siemens').

### property delta\_values

Get delta sense readings stored in 6221 buffer.

### disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

# disable\_output\_trigger()

Disables the output trigger for the Trigger layer

# disable\_source()

Disables the source of current or voltage depending on the configuration of the instrument.

# property display\_enabled

Control (boolean) whether or not the display of the sourcemeter is enabled. Valid values are True and False.

### enable\_source()

Enables the source of current or voltage depending on the configuration of the instrument.

# property error

Get the next error from the queue.

Deprecated since version 0.15: Use *next\_error* instead.

### property id

Get the identification of the instrument.

### is\_buffer\_full()

Return True if the buffer is full of measurements.

#### property measurement\_event\_enabled

Control which measurement events are registered in the Measurement Summary Bit (MSB) status bit. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits.

#### property measurement\_events

Get which measurement events have been registered in the Measurement event registers. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits. Reading this value clears the register.

### property next\_error

Get the next error in the queue. If you want to read and log all errors, use *check\_errors()* instead.

# property operation\_event\_enabled

Control which operation events are registered in the Operation Summary Bit (OSB) status bit. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits.

# property operation\_events

Get which operation events have been registered in the Operation event registers. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits. Reading this value clears the register.

### property options

Get the device options installed.

# property output\_low\_grounded

Control (boolean) whether the low output of the triax connection is connected to earth ground (True) or is floating (False).

### output\_trigger\_on\_external(line=1, after='DEL')

Configures the output trigger on the specified trigger link line number, with the option of supplying the part of the measurement after which the trigger should be generated (default to delay, which is right before the measurement)

# **Parameters**

- line A trigger line from 1 to 4
- **after** An event string that determines when to trigger

### property questionable\_event\_enabled

Control which questionable events are registered in the Questionable Summary Bit (QSB) status bit. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits.

# property questionable\_events

Get which questionable events have been registered in the Questionable event registers. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits. Reading this value clears the register.

# read(\*\*kwargs)

Read up to (excluding) *read\_termination* or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Resets the instrument and clears the queue.

# reset\_buffer()

Reset the buffer.

### set\_timed\_arm(interval)

Sets up the measurement to be taken with the internal trigger at a variable sampling rate defined by the interval in seconds between sampling points

# property shield\_to\_guard\_enabled

Control if shield is connected to the guard(boolean).

If not, the shield is connected to the output-low.

### shutdown()

Disables the output.

### property source\_auto\_range

Control (boolean) the auto range of the current source. Valid values are True or False.

# property source\_compliance

Control (floating) the compliance of the current source in Volts. valid values are in range 0.1 [V] to 105 [V].

# property source\_current

Control (floating) the source current in Amps.

### property source\_delay

Control (floating) a manual delay for the source after the output is turned on before a measurement is taken. When this property is set, the auto delay is turned off. Valid values are between 1e-3 [seconds] and 999999.999 [seconds].

# property source\_enabled

Control (boolean) whether the source is enabled, takes values True or False. The convenience methods <code>enable\_source()</code> and <code>disable\_source()</code> can also be used.

# property source\_range

Control (floating) the source current range in Amps, which can take values between -0.105 A and +0.105 A. Auto-range is disabled when this property is set.

# property srq\_event\_enabled

Control which event registers trigger the Service Request (SRQ) status bit. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits.

### property standard\_event\_enabled

Control which standard events are registered in the Event Summary Bit (ESB) status bit. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits.

### property standard\_events

Get which standard events have been registered in the Standard event registers. Refer to the Model 6220/6221 Reference Manual for more information about programming the status bits. Reading this value clears the register.

# start\_buffer()

Starts the buffer.

### property status

Get the status byte and Master Summary Status bit.

# stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

# triad(base\_frequency, duration)

Sounds a musical triad using the system beep.

#### **Parameters**

- base\_frequency A frequency in Hz between 65 Hz and 1.3 MHz
- **duration** A time in seconds between 0 and 7.9 seconds

### trigger()

Executes a bus trigger, which can be used when trigger\_on\_bus() is configured.

# trigger\_immediately()

Configures measurements to be taken with the internal trigger at the maximum sampling rate.

# trigger\_on\_bus()

Configures the trigger to detect events based on the bus trigger, which can be activated by trigger().

### trigger\_on\_external(line=1)

Configures the measurement trigger to be taken from a specific line of an external trigger

### **Parameters**

line – A trigger line from 1 to 4

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

# **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

# wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

### **Parameters**

- **should\_stop** A function that returns True when this function should return early
- **timeout** A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

### waveform\_abort()

Abort the waveform output and disarm the waveform function.

# property waveform\_amplitude

Control (floating) the (peak) amplitude of the waveform in Amps. Valid values are in range 2e-12 to 0.105.

### waveform\_arm()

Arm the current waveform function.

### property waveform\_duration\_cycles

Control (floating) the duration of the waveform in cycles. Valid values are in range 1e-3 to 9999999900.

# waveform\_duration\_set\_infinity()

Set the waveform duration to infinity.

# property waveform\_duration\_time

Control (floating) the duration of the waveform in seconds. Valid values are in range 100e-9 to 999999.999.

# property waveform\_dutycycle

Control (floating) the duty-cycle of the waveform in percent for the square and ramp waves. Valid values are in range 0 to 100.

# property waveform\_frequency

Control (floating) the frequency of the waveform in Hertz. Valid values are in range 1e-3 to 1e5.

# property waveform\_function

Control (string) the selected wave function. Valid values are "sine", "ramp", "square", "arbitrary1", "arbitrary2", "arbitrary3" and "arbitrary4".

# property waveform\_offset

Control (floating) the offset of the waveform in Amps. Valid values are in range -0.105 to 0.105.

# property waveform\_phasemarker\_line

Control (numerical) the line of the phase marker.

### property waveform\_phasemarker\_phase

Control (numerical) the phase of the phase marker.

# property waveform\_ranging

Control (string) the source ranging of the waveform. Valid values are "best" and "fixed".

### waveform\_start()

Start the waveform output. Must already be armed

### property waveform\_use\_phasemarker

Control (boolean) whether the phase marker option is turned on or of. Valid values True (on) or False (off). Other settings for the phase marker have not yet been implemented.

# write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

### **Parameters**

- **command** Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.31.13 Keithley 6517B Electrometer

Bases: KeithleyBuffer, SCPIMixin, Instrument

Represents the Keithley 6517B ElectroMeter and provides a high-level interface for interacting with the instrument.

```
keithley = Keithley6517B("GPIB::1")
keithley.apply_voltage()
                                     # Sets up to source current
keithley.source_voltage_range = 200 # Sets the source voltage
                                     # range to 200 V
keithley.source_voltage = 20
                                     # Sets the source voltage to 20 V
keithley.enable_source()
                                     # Enables the source output
                                     # Sets up to measure resistance
keithley.measure_resistance()
                                      # Ramps the voltage to 50 V
keithley.ramp_to_voltage(50)
print(keithley.resistance)
                                      # Prints the resistance in Ohms
                                      # Ramps the voltage to 0 V
keithley.shutdown()
                                      # and disables output
```

# apply\_voltage(voltage\_range=None)

Configures the instrument to apply a source voltage, and uses an auto range unless a voltage range is specified.

#### **Parameters**

```
voltage_range – A voltage_range value or None (activates auto range)
```

### auto\_range\_source()

Configures the source to use an automatic range.

# property buffer\_data

Get a numpy array of values from the buffer.

### property buffer\_points

Control (integer) the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

# check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### **Returns**

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

### **Parameters**

- **points** The number of points in the buffer.
- **delay** The delay time in seconds.

# property current

Get the current in Amps, if configured for this reading.

# property current\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC current measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property current\_range

Control (floating) the measurement current range in Amps, which can take values between -20 and +20 mA. Auto-range is disabled when this property is set.

# disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

# disable\_source()

Disables the source of current or voltage depending on the configuration of the instrument.

### enable\_source()

Enables the source of current or voltage depending on the configuration of the instrument.

### property error

Get the next error from the queue.

Deprecated since version 0.15: Use next\_error instead.

### static extract\_value(result)

extracts the physical value from a result object returned by the instrument

### property id

Get the identification of the instrument.

### is\_buffer\_full()

Return True if the buffer is full of measurements.

# measure\_current(nplc=1, current=0.000105, auto\_range=True)

Configures the measurement of current.

### **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- current Upper limit of current in Amps, from -21 mA to 21 mA
- auto\_range Enables auto\_range if True, else uses the current\_range attribute

measure\_resistance(nplc=1, resistance=210000.0, auto\_range=True)

Configures the measurement of resistance.

### **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- resistance Upper limit of resistance in Ohms, from -210 POhms to 210 POhms
- auto\_range Enables auto\_range if True, else uses the resistance\_range attribute

**measure\_voltage**(*nplc=1*, *voltage=21.0*, *auto range=True*)

Configures the measurement of voltage.

### **Parameters**

- nplc Number of power line cycles (NPLC) from 0.01 to 10
- voltage Upper limit of voltage in Volts, from -1000 V to 1000 V
- auto\_range Enables auto\_range if True, else uses the voltage\_range attribute

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# property options

Get the device options installed.

### ramp\_to\_voltage(target\_voltage, steps=30, pause=0.02)

Ramps to a target voltage from the set voltage value over a certain number of linear steps, each separated by a pause duration.

# **Parameters**

- target\_voltage A voltage in Volts
- **steps** An integer number of steps
- pause A pause duration in seconds to wait between steps

# read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Resets the instrument and clears the queue.

### reset\_buffer()

Reset the buffer.

### property resistance

Get the resistance in Ohms, if configured for this reading.

# property resistance\_nplc

Control (floating) the number of power line cycles (NPLC) for the 2-wire resistance measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

### property resistance\_range

Control (floating) the resistance range in Ohms, which can take values from 0 to 100e18 Ohms. Auto-range is disabled when this property is set.

### shutdown()

Ensures that the current or voltage is turned to zero and disables the output.

# property source\_current\_resistance\_limit

Control whether the current limit is enabled.

# property source\_enabled

Get whether the source is enabled.

# property source\_voltage

Control (floating) the source voltage in Volts.

# property source\_voltage\_range

Control (floating) the source voltage range in Volts, which can take values from -1000 to 1000 V. Auto-range is disabled when this property is set.

### start\_buffer()

Starts the buffer.

# property status

Get the status byte and Master Summary Status bit.

# stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

#### trigger()

Executes a bus trigger, which can be used when trigger\_on\_bus() is configured.

# trigger\_immediately()

Configures measurements to be taken with the internal trigger at the maximum sampling rate.

### trigger\_on\_bus()

Configures the trigger to detect events based on the bus trigger, which can be activated by trigger().

# property voltage

Get the voltage in Volts, if configured for this reading.

# property voltage\_nplc

Control (floating) the number of power line cycles (NPLC) for the DC voltage measurements, which sets the integration period and measurement speed. Takes values from 0.01 to 10, where 0.1, 1, and 10 are Fast, Medium, and Slow respectively.

# property voltage\_range

Control (floating) the measurement voltage range in Volts, which can take values from -1000 to 1000 V. Auto-range is disabled when this property is set.

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

### wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

#### **Parameters**

- **should\_stop** A function that returns True when this function should return early
- timeout A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

### write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

### **Parameters**

- **command** Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.31.14 Keithley DMM6500 Multimeter

```
Bases: SCPIMixin, Instrument
```

Represent the Keithely DMM6500 6½-Digit Multimeter and provide a high-level interface for interacting with the instrument. This class only uses "SCPI" command set (see also *command\_set*) to communicate with the instrument.

```
# Access via LAN
ip_address = "xxx.xxx.xxx.xxx"
dmm = KeithleyDMM6500(f"TCPIP::{ip_address}::inst0::INSTR")
```

User can also use PyVISA to get DMM6500's USB port name and pass it to KeithleyDMM6500

```
import pyvisa
rm = pyvisa.ResourceManager()
resources = rm.list_resources()

# assume there is only one USB instruments
# Ex. ('USB0::1510::25856::01234567::0::INSTR')

dmm = KeithleyDMM6500( resources[0] )

# Measure voltage
dmm.measure_voltage()
print(dmm.voltage)

# Measure AC voltage
dmm.measure_voltage(ac=True)
print(dmm.voltage)
```

# channels

# Channels

```
ScannerCard2000Channel.
ch 1:
          ScannerCard2000Channel.
                                    ch_2:
ch 3:
                                    ch 4:
                                               ScannerCard2000Channel.
          ScannerCard2000Channel.
ch 5:
          ScannerCard2000Channel,
                                    ch 6:
                                               ScannerCard2000Channel,
ch 7:
       ScannerCard2000Channel, ch_8:
                                        ScannerCard2000Channel, ch_9:
ScannerCard2000Channel, ch_10: ScannerCard2000Channel
```

### acquire\_relative(mode=None)

Set the active value as the relative for the active mode, or can set another mode by its name.

#### **Parameters**

**mode** – A valid *mode* name, or *None* for the active mode

#### Returns

The relative value that was acquired

### property aperture

Control the aperture time of currently active *mode*.

Valid values: MIN, DEF, MAX, or number between 8.333u and 0.25 s.

# 1 Note

Only voltage, current, resistance, resistance 4W, diode, temperature, frequency, period, and voltage ratio mode support aperture setting. If current active mode doesn't support aperture, this command will hang till adapter's timeout and cause -113 "Undefined header" error.

# auto\_range(mode=None)

Set the active mode to use auto-range, or can set another mode by its name.

Only current (DC), current ac, voltage (DC), voltage ac, resistance (2-wire), resistance 4W (4-wire), capacitance, and voltage ratio support autorange. If chosen mode is not in these modes, this command will do nothing.

#### **Parameters**

**mode** – A valid *mode* name, or *None* for the active mode

# auto\_range\_status(mode=None)

Get the status of auto-range of active mode or another mode by its name. Only current (DC), current ac, voltage (DC), voltage ac, resistance (2-wire), resistance 4W (4-wire), capacitance, and voltage ratio support autorange. If chosen mode is not in these modes, this command will also return False.

### **Parameters**

**mode** – A valid *mode* name, or *None* for the active mode

### Returns

a bool value for auto-range enabled or disabled

### Return type

bool

# property autorange\_enabled

Control the autorange state for currently active *mode*.



# 1 Note

If currently active mode doesn't support autorange, this command will hang till adapter's timeout and cause -113 "Undefined header" error.

# property autozero\_enabled

Control automatic updates to the internal reference measurements (autozero) of the instrument.

### **beep**(*frequency*, *duration*)

Sound a system beep.

#### **Parameters**

- **frequency** A frequency in Hz between 20 Hz and 8000 Hz
- duration The amount of time to play the tone between 0.001 s to 100 s

### **Returns**

None

### property buffer\_points

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead. 0 means the largest buffer possible based on the available memory when the bufer is created.

### property buffer\_size

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead. 0 means the largest buffer possible based on the available memory when the bufer is created.

# property capacitance

Measure a capacitance in Farad, based on the active *mode*.

# property capacitance\_digits

Control the number of digits in the capacitance readings (integer strictly from 3 to 6). See also the digits.

# property capacitance\_range

Control the capacitance full-scale measure range in Farad. Available ranges are 1e-9, 10e-9, 100e-9, 1e-6, 10e-6, 100e-6, 1e-3. Auto-range is disabled when this property is set. See also the *range*.

# property capacitance\_relative

Control the capacitance relative value in Farad (float strictly from -0.001 to 0.001 F). See also the relative.

# property capacitance\_relative\_status

Control a relative offset value applied to capacitance measurement. See also the relative\_enabled.

#### close()

Close the connection

# property command\_set

Control the command set that to use with DMM6500. Reboot the instrument is needed after changing the command set. Available values are: SCPI, TSP, SCPI2000, and SCPI34401. The KeithleyDMM6500 class was designed to use SCPI command set only.



# Mote

If you want to use TSP command set, you can use write() and ask() to send TSP command instead.

# property current

Measure a DC or AC current in Amps, based on the active mode.

# property current\_ac\_bandwidth

Control the detector bandwidth in Hz for AC current measurement (integer strictly among 3, 30, and 300).

# property current\_ac\_digits

Control the number of digits in the AC current readings (integer strictly from 3 to 6). See also the digits.

#### property current\_ac\_range

Control the AC current positive full-scale measure range in Amps. Available ranges are 1e-3, 10e-3, 100e-3, 1, 3 Amps (for front terminals), and 10 Amps (for rear terminals). See also the *range*.

#### property current\_ac\_relative

Control the AC current relative value in Amps (float strictly from -3 to 3). See also the *relative*.

# property current\_ac\_relative\_enabled

Control a relative offset value applied to AC current measurement. See also the relative\_enabled.

# property current\_digits

Control the number of digits in the DC current readings (integer strictly from 3 to 6). See also the digits.

### property current\_nplc

Control the number of power line cycles (NPLC) for the DC current measurement (float strictly from 0.0005 to 15). See also the *nplc*.

# property current\_range

Control the DC current full-scale measure range in Amps. Available ranges are 10e-6, 100e-6, 1e-3, 10e-3, 100e-3, 1, 3 Amps (for front terminals), and 10 Amps (for rear terminals). Auto-range is disabled when this property is set. See also the *range*.

### property current\_relative

Control the DC current relative value in Amps (float strictly from -3 to 3). See also the *relative*.

### property current\_relative\_enabled

Control a relative offset value applied to DC current measurement. See also the relative\_enabled.

#### property data\_format

Control data format that is used when transferring readings over the remote interface. Available values are ASC (ASCII), REAL (double-precision), or SRE (single-precision).

# property detector\_bandwidth

Control the lowest frequency expected in the input signal in Hz ONLY for AC voltage and AC current measurement,

Valid values: 3, 30, 300, MIN, DEF, MAX.

# property digits

Control the displaying number of digits for currently active *mode*. Available values are from 3 to 6 representing display digits from 3.5 to 6.5.

# property diode

Measure a diode's forward voltage drop of general-purpose diodes and the Zener voltage of Zener diodes on the 10V range with a constant test current (bias level), based on the active *mode*.

# property diode\_bias

Control the amount of current in Amps the instrument sources while making measurement. Available bias levels are 1e-5, 0.0001, 0.001, 0.001.

### property diode\_nplc

Control the number of power line cycles (NPLC) for the diode measurement (float strictly from 0.0005 to 15). See also the nplc.

### disable\_filter(mode=None)

Disable the averaging filter for the active mode, or can set another mode by its name.

#### **Parameters**

mode – A valid mode name, or None for the active mode

### Returns

Filter status read from the instrument

### disable\_relative(mode=None)

Disable the application of a relative offset value to the measurement for the active mode, or can set another mode by its name.

#### **Parameters**

mode – A valid mode name, or None for the active mode

# property display\_screen

Set displayed front-panel screen by the name. Available names are: HOME (home), HOME\_LARG (home screen with large readings), READ (reading table), HIST (histogram), SWIPE\_FUNC (FUNCTIONS swipe screen), SWIPE\_GRAP (GRAPH swipe screen), SWIPE\_SEC (SECONDARY swipe screen), SWIPE\_SETT (SETTINGS swipe screen), SWIPE\_STAT (STATISTICS swipe screen), SWIPE\_USER (USER swipe screen), SWIPE\_CHAN (CHANNEL swipe screen), SWIPE\_NONS (NONSWITCH swipe screen), SWIPE\_SCAN (SCAN swipe screen), CHANNEL\_CONT (Channel control screen), CHANNEL\_SETT (Channel settings screen), CHANNEL\_SCAN (Channel scan screen), or PROC (minimal CPU resources).

# displayed\_text(top line=None, bot line=None)

Display text messages on the front-panel USER swipe screen. If no messages were defined, screen will be cleared.

#### **Parameters**

- top\_line 1st line message
- bot\_line 2nd line message

### Returns

None

# enable\_filter(mode=None, type='repeat', count=1)

Enable the averaging filter for the active mode, or can set another mode by its name.

#### **Parameters**

- mode A valid mode name, or None for the active mode
- **type** The type of averaging filter, could be REPeat, MOVing, or HYBRid.
- **count** A number of averages, which can take take values from 1 to 100

#### Returns

Filter status read from the instrument

# enable\_relative(mode=None)

Enable the application of a relative offset value to the measurement for the active mode, or can set another mode by its name.

### **Parameters**

**mode** – A valid *mode* name, or *None* for the active mode

# property frequency

Measure a frequency in Hz, based on the active *mode*.

### property frequency\_aperature

Control the aperture time in seconds for frequency measurement (float strictly from 2 ms to 273 ms). See also *aperture*.

### property frequency\_digits

Control the number of digits in the frequency readings (integer strictly from 3 to 6). See also the digits.

# property frequency\_relative

Control the frequency relative value in Hz (float strictly from -1 MHz to 1 MHz). See also the *relative*.

# property frequency\_relative\_enabled

Control a relative offset value applied to frequency measurement. See also the relative\_enabled.

### property frequency\_threshold

Control the expected input level in Volts for the frequency measurement (float strictly from 0.1 to 750V).

# property frequency\_threshold\_auto\_enabled

Control the auto threshold range for frequency measurement enabled or not.

# property line\_frequency

Get the power line frequency which automatically detected while the instrument is powered on.

# measure\_capacitance(max\_capacitance=0.001)

Configure the instrument to measure capacitance.

### **Parameters**

```
max_capacitance - Set capacitance_range after changing mode
```

#### Returns

None

### measure\_continuity()

Configure the instrument to perform continuity testing.

# Returns

None

# measure\_current(max\_current=0.01, ac=False)

Configure the instrument to measure current, based on a maximum current to set the range, and a boolean flag to determine if DC or AC is required.

### **Parameters**

- max\_current A current in Volts to set the current range
- ac False for DC current, and True for AC current

### measure\_diode()

Configure the instrument to perform diode testing.

### Returns

None

# measure\_frequency()

Configure the instrument to measure frequency.

# measure\_period()

Configure the instrument to measure period.

# measure\_resistance(max\_resistance=10000000.0, wires=2)

Configure the instrument to measure resistance, based on a maximum resistance to set the range.

### **Parameters**

• max\_resistance (float) - A resistance in Ohms to set the resistance range

• wires (int) – 2 for normal resistance, and 4 for 4-wires resistance

#### Returns

None

# measure\_temperature()

Configure the instrument to measure temperature.

# measure\_voltage(max\_voltage=1, ac=False)

Configure the instrument to measure voltage, based on a maximum voltage to set the range, and a boolean flag to determine if DC or AC is required.

### **Parameters**

- max\_voltage A voltage in Volts to set the voltage range
- ac False for DC voltage, and True for AC voltage

# property mode

Control the active measure function. Available values are: current (DC), current ac, voltage (DC), voltage ac, resistance (2-wire), resistance 4W (4-wire), diode, capacitance, temperature, continuity, period, frequency, and voltage ratio.

### property nplc

Control the integration time in number of power line cycles (NPLC) of currently active mode. This value sets the amount of time that the input signal is measured. Valid values are: 0.0005 to 15 (60Hz) or 12 (50Hz or 400Hz).



### Note

Only voltage, current, resistance, resistance 4W, diode, temperature, and voltage ratio mode support NPLC setting. If current active mode doesn't support NPLC, this command will hang till adapter's timeout and cause -113 "Undefined header" error.

### property period

Measure a period in seconds, based on the active *mode*.

# property period\_aperature

Control the aperture time in seconds for period measurement (float strictly from 2 ms to 273 ms). See also aperture

# property period\_digits

Control the number of digits in the period readings (integer strictly from 3 to 6). See also the digits.

# property period\_relative

Control the period relative value in seconds (float strictly from -1 s to 1 s). See also the *relative*.

# property period\_relative\_enabled

Control a relative offset value applied to period measurement. See also the relative\_enabled.

### property period\_threshold

Control the expected input level in Volts for the period measurement (float strictly from 0.1 to 750V).

# property period\_threshold\_auto\_enabled

Control the auto threshold range for period measurement enabled or not.

### property points\_in\_buffer

Get the number of readings stored in the buffer.

### property pseudo\_scanner\_enabled

Set pseudo scanner card if there's no scanner card in the instrument. After setting, user can check current scanner card by <code>scan\_id</code>. If a scanner card is installed, this setting won't have any effect.

### property range

Control the positive full-scale measure range for currently active *mode*. Auto-range is disabled when this property is set.

For frequency and period measurements, ranging applies to the signal's input voltage, not its frequency

### property relative

Control the relative offset value of currently active *mode*. When relative offset is enabled, all subsequent measured readings are offset by the value that is set for this command. If the instrument acquires the value, read this setting to return the value that was measured internally. See also the *relative\_enabled*.

# property relative\_enabled

Control the relative offset value applied to new measurements for currently active *mode*.

### property resistance

Measure a resistance in Ohms for both 2-wire and 4-wire configurations, based on the active mode.

# property resistance\_4W\_digits

Control the number of digits in the 4-wire resistance readings (integer strictly from 3 to 6). See also the *digits*.

# property resistance\_4W\_nplc

Control the number of power line cycles (NPLC) for the 4-wire resistance measurement (float strictly from 0.0005 to 15). See also the *np1c*.

# property resistance\_4W\_range

Control the 4-wire resistance full-scale measure range in Ohms. Available ranges are: 1, 10, 100, 1e3, 10e3, 10e3, 1e6, 10e6, and 100e6. Auto-range is disabled when this property is set. See also the *range*.

# property resistance\_4W\_relative

Control the 4-wire resistance relative value in Ohms (float strictly from -100M to 100M). See also the *relative*.

# property resistance\_4W\_relative\_enabled

Control a relative offset value applied to 4-wire resistance measurement. See also the relative\_enabled.

### property resistance\_digits

Control the number of digits in the 2-wire resistance readings (integer strictly from 3 to 6). See also the *digits*.

# property resistance\_nplc

Control the number of power line cycles (NPLC) for the 2-wire resistance measurement (float strictly from 0.0005 to 15). See also the *np1c*.

# property resistance\_range

Control the 2-wire resistance full-scale measure range in Ohms. Available ranges are: 10, 100, 1e3, 10e3, 10e3, 1e6, 10e6, and 100e6. Auto-range is disabled when this property is set. See also the *range*.

#### property resistance\_relative

Control the 2-wire resistance relative value in Ohms (float strictly from -100M to 100M). See also the *relative*.

### property resistance\_relative\_enabled

Control a relative offset value applied to 2-wire resistance measurement. See also the relative\_enabled.

# property scan\_card\_vmax

Get the maximum voltage of all channels.

### property scan\_channels

Control the channel list of scanning. An empty string will clear the list. Use comma to separate single channel and use a colon to separate the first and last channel in the list. Examples: 1, 1, 3, 5, 1:2, 7:8, or 1:10.

### property scan\_channels\_list

Get scan\_channels string to a list of integers.

For example, when *scan\_channels* is 1,3:5,7:8,10, this attribute will return [1,3,4,5,7,8,10]. If scan\_channels\_list=[1,2,3,4,6], the *scan\_channels* will be 1:4,6.

### property scan\_count

Control the number of times the scan is repeated. Set to 0 set the scan to repeat until aborted.

# property scan\_id

Get scanner card's ID.

### property scan\_interval

Control the interval time (0s to 100ks) between scan starts when the *scan\_count* is more than one.

# property scan\_iscomplete

Get Event Status Register (ESR) bit 0 to determine if previous works were completed. This property is used while running time-consuming scanning operation.

# property scan\_modes

Get a dictionary of every channel's mode.

# scan\_start(block\_communication=True, count=None, interval=None)

Start the scanner card to close each channel of scan\_channels sequentially and to do measurements.

If <code>scan\_count</code> is larger than 1, the next scanning will start again after <code>scan\_interval</code> second. Running large counts or long interval scanning is a time-consuming operation. It's better to set <code>block\_communication=False</code> and use <code>scan\_iscomplete</code> to check if the measurement is completed.

### **Parameters**

- **block\_communication** A bool value which controls communication state while scanning. Default is True and the communication waits until the commands are complete to accept new commands
- **count** An alternative way to set *scan\_count* before scanning.
- **interval** An alternative way to set *scan\_interval* in second before scanning.

### Returns

None

### scan\_stop()

Abort the scanning measurement by stopping the measurement arming and triggering sequence.

# Returns

None

### property scan\_vch\_end

Get the last channel in the slot that supports voltage or 2-wire measurements.

# property scan\_vch\_start

Get the first channel in the slot that supports voltage or 2-wire measurements.

### **scanned\_data**(*start\_idx=None*, *end\_idx=None*, *raw=False*)

Return a list of scanning values from the buffer.

### **Parameters**

- start\_idx A bool value which controls communication state while scanning. Default
  is True and the communication waits until the commands are complete to accept new
  commands
- end\_idx An alternative way to set scan\_count
- raw An alternative way to set scan\_interval in second

#### Returns

A list of scan channels' measured

### Return type

A list of channels' list

### property system\_time

Control system time on the instrument. Format of set is: year, month, day, hour, minute, second or hour, minute, second. Example: Using time package to set instrument's clock: dmm. system\_time = time.strftime("%Y, %m, %d, %H, %M, %S")

### property temperature

Measure a temperature in Celsius, based on the active *mode*.

# property temperature\_digits

Control the number of digits in the temperature readings (integer strictly from 3 to 6). See also the digits.

### property temperature\_nplc

Control the number of power line cycles (NPLC) for the temperature measurement (float strictly from 0.0005 to 15). See also the nplc.

# property temperature\_relative

Control the temperature relative value in Celsius (float strictly from -3310 C to 3310 C). See also the *relative*.

# property temperature\_relative\_enabled

Control a relative offset value applied to temperature measurement. See also the relative\_enabled.

# property terminals\_used

Get which set of input and output terminals the instrument is using. Return can be FRONT or REAR.

# trigger\_single\_autozero()

Cause the instrument to refresh the reference and zero measurements once.

Consequent autozero measurements are disabled.

# property voltage

Measure a DC or AC voltage in Volts, based on the active mode.

# property voltage\_ac\_bandwidth

Control the detector bandwidth in Hz for AC voltage measurement (integer strictly among 3, 30, and 300).

### property voltage\_ac\_digits

Control the number of digits in the AC voltage readings (integer strictly from 3 to 6). See also the digits.

# property voltage\_ac\_range

Control the AC voltage positive full-scale measure range in Volts. Available ranges are 0.1, 1, 10, 100, 750. Auto-range is disabled when this property is set. See also the *range*.

# property voltage\_ac\_relative

Control the AC voltage relative value in Volts (float strictly from -750 to 750). See also the relative.

# property voltage\_ac\_relative\_enabled

Control a relative offset value applied to AC voltage measurement. See also the relative\_enabled.

# property voltage\_digits

Control the number of digits in the DC voltage readings (integer strictly from 3 to 6). See also the digits.

### property voltage\_nplc

Control the number of power line cycles (NPLC) for the DC voltage measurement (float strictly from 0.0005 to 15). See also the *nplc*.

### property voltage\_range

Control the DC voltage full-scale measure range in Volts. Available ranges are 0.1, 1, 10, 100, 1000. Autorange is disabled when this property is set. See also the *range*.

# property voltage\_relative

Control the DC voltage relative value in Volts (float strictly from -1000 to 1000). See also the relative.

# property voltage\_relative\_enabled

Control a relative offset value applied to DC voltage measurement. See also the relative\_enabled.

**class** pymeasure.instruments.keithley.keithleyDMM6500.**ScannerCard2000Channel**(*parent*, *id*)

Bases: Channel

# property autorange\_enabled

Control the autorange state for currently active mode.



# 1 Note

If current active mode doesn't support autorange, this command will hang till adapter's timeout and cause -113 "Undefined header" error.

### disable filter(mode=None)

Disable the averaging filter for the active mode, or can set another mode by its name.

### **Parameters**

**mode** – A valid *mode* name, or *None* for the active mode

# Returns

Filter status read from the instrument

# enable\_filter(mode=None, type='repeat', count=1)

Enable the averaging filter for the active mode, or can set another mode by its name.

### **Parameters**

- mode A valid mode name, or *None* for the active mode
- type The type of averaging filter, could be REPeat, MOVing, or HYBRid.

• count – A number of averages, which can take take values from 1 to 100

#### Returns

Filter status read from the instrument

# property mode

Control the configuration mode for measurements, which can take the values: current (DC), current ac, voltage (DC), voltage ac, resistance (2-wire), resistance 4W (4-wire), diode, capacitance, temperature, continuity, period, frequency, and voltage ratio.

# property nplc

Control the integration time in number of power line cycles (NPLC). Valid values: 0.0005 to 15 (60Hz) or 12 (50Hz or 400Hz) This command is valid only for voltage, 2-wire ohms, and 4-wire ohms.



Only voltage, current, resistance, resistance 4W, diode, temperature, and voltage ratio mode support NPLC setting. If current active mode doesn't support NPLC, this command will hang till adapter's timeout and cause -113 "Undefined header" error.

# property range

Control measuring range for currently active mode. For frequency and period measurements, *range* applies to the signal's input voltage, not its frequency

# write(command)

Write a command to the instrument.

# 7.31.15 Keithley 2182 Nanovoltmeter

Bases: SCPIMixin, KeithleyBuffer, Instrument

Represents the Keithley 2182 Nanovoltmeter and provides a high-level interface for interacting with the instrument.

```
keithley = Keithley2182("GPIB::1")
keithley.reset()
                                        # Return instrument settings to default
                                          values
keithley.thermocouple = 'S'
                                        # Sets thermocouple type to S
keithley.active_channel = 1
                                        # Sets channel 1 for active measurement
keithley.channel_function = 'voltage' # Configures active channel for voltage
                                          measurement
print(keithley.voltage)
                                        # Prints the voltage in volts
keithley.ch_1.setup_voltage()
                                        # Set channel 1 active and prepare
                                          voltage measurement
keithley.ch_2.setup_temperature()
                                        # Set channel 2 active and prepare
                                          temperature measurement
```

ch\_1

### Channel

Keithley2182Channel

### ch\_2

#### Channel

Keithley2182Channel

# property active\_channel

Control which channel is active for measurement. Valid values are 0 (internal temperature sensor), 1, and 2.

# auto\_line\_frequency()

Set appropriate limits for NPLC voltage and temperature readings.

### property auto\_zero\_enabled

Control the auto zero option (bool).

# property buffer\_data

Get a numpy array of values from the buffer.

# property buffer\_points

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

# property channel\_function

Control the measurement mode of the active channel. Valid options are voltage and temperature.

### check\_errors()

Read all errors from the instrument.

### Returns

List of error entries.

# check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

# Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

### clear()

Clear the instrument status byte.

### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

### **Parameters**

- points The number of points in the buffer.
- **delay** The delay time in seconds.

### disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

# property display\_enabled

Control whether the front display of the voltmeter is enabled. Valid values are True and False.

### property id

Get the identification of the instrument.

# property internal\_temperature

Measure the internal temperature in Celsius.

### is\_buffer\_full()

Return True if the buffer is full of measurements.

# property line\_frequency

Get the line frequency in Hertz. Values are 50 or 60. Cannot be set on 2182.

### property maximum

Get the calculated maximum from the buffer data.

### property mean

Get the calculated mean (average) from the buffer data.

# property minimum

Get the calculated minimum from the buffer data.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

### property options

Get the device options installed.

### read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument and clear the queue.

### reset\_buffer()

Reset the buffer.

# sample\_continuously()

Configure the instrument to continuously read samples and turn off any buffer or output triggering.

### shutdown()

Brings the instrument to a safe and stable state

### property standard\_dev

Get the calculated standard deviation from the buffer data.

### start\_buffer()

Starts the buffer.

# property status

Get the status byte and Master Summary Status bit.

### stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

# property temperature

Measure the temperature in Celsius, if active channel is configured for this reading.

### property temperature\_nplc

Control the number of power line cycles (NPLC) for temperature measurements, which sets the integration period and measurement speed. Valid values are from 0.01 to 50 or 60, depending on the line frequency. Default is 5.(dynamic)

# property temperature\_reference\_junction

Control whether the thermocouple reference junction is internal (INT) or simulated (SIM). Default is INT.

### property temperature\_simulated\_reference

Control the value of the simulated thermocouple reference junction in Celsius. Default is 23 C.

# property thermocouple

Control the thermocouple type for temperature measurements. Valid options are B, E, J, K, N, R, S, and T.

# trigger()

Execute a bus trigger, which can be used when trigger\_on\_bus() is configured.

### property trigger\_count

Control the trigger count which can take values from 1 to 9,999. Default is 1.

# property trigger\_delay

Control the trigger delay in seconds, which can take values from 0 to 999999.999 s. Default is 0.

### trigger\_immediately()

Configure measurements to be taken with the internal trigger at the maximum sampling rate.

# trigger\_on\_bus()

Configure the trigger to detect events based on the bus trigger, which can be activated by trigger().

### property voltage

Measure the voltage in Volts, if active channel is configured for this reading.

### property voltage\_nplc

Control the number of power line cycles (NPLC) for voltage measurements, which sets the integration period and measurement speed. Valid values are from 0.01 to 50 or 60, depending on the line frequency. Default is 5.(dynamic)

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
wait_for_buffer(should_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)
```

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

#### Parameters

- **should\_stop** A function that returns True when this function should return early
- timeout A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

# write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# class pymeasure.instruments.keithley.keithley2182.Keithley2182Channel(parent, id)

Bases: Channel

Implementation of a Keithley 2182 channel.

Channel 1 is the fundamental measurement channel, while channel 2 provides sense measurements. Channel 2 inputs are referenced to Channel 1 LO. Possible configurations are

Voltage (Channel 1) Temperature (Channel 1) Voltage (Channel 1) and Voltage (Channel 2) Voltage (Channel 1) and Temperature (Channel 2)

# acquire\_temperature\_reference()

Acquire a temperature measurement and store it as the relative offset value.

Only acquires reference if temperature offset is enabled.

# acquire\_voltage\_reference()

Acquire a voltage measurement and store it as the relative offset value.

Only acquires reference if voltage offset is enabled.

# setup\_temperature(nplc=5)

Change active channel and configure channel for temperature measurement.

#### **Parameters**

**nplc** – Number of power line cycles (NPLC) from 0.01 to 50/60

# setup\_voltage(auto\_range=True, nplc=5)

Set active channel and configure channel for voltage measurement.

#### **Parameters**

- auto\_range Enables auto\_range if True, else uses set voltage range
- nplc Number of power line cycles (NPLC) from 0.01 to 50/60

# property temperature\_offset

Control the relative offset for measuring temperature. Displayed value = actual value - offset value. Valid values are -273 C to 1800 C.

# property temperature\_offset\_enabled

Control whether temperature is measured as a relative or absolute value (bool). Disabled by default.

# property voltage\_offset

Control the relative offset for measuring voltage. Displayed value = actual value - offset value. Valid ranges are -120 V to +120 V for Ch. 1, and -12 V to +12 V for Ch. 2.(dynamic)

### property voltage\_offset\_enabled

Control whether voltage is measured as a relative or absolute value (bool). Enabled by default for Ch. 2 voltage, which is measured relative to Ch. 1 voltage.

# property voltage\_range

Control the positive full-scale measurement voltage range in Volts. The Keithley 2182 selects a measurement range based on the expected voltage. DCV1 has five ranges: 10 mV, 100 mV, 1 V, 10 V, and 100 V. DCV2 has three ranges: 100 mV, 1 V, and 10 V. Valid limits are from 0 to 120 V for Ch. 1, and 0 to 12 V for Ch. 2. Auto-range is automatically disabled when this property is set.(dynamic)

# property voltage\_range\_auto\_enabled

Control the auto voltage ranging option (bool).

# 7.31.16 Keithley DAQ6510 Data Acquisition Logging Multimeter System

Bases: KeithleyBuffer, SCPIMixin, Instrument

Represents the Keithley DAQ6510 Data Acquisition Logging Multimeter System and provides a high-level interface for interacting with the instrument.

```
keithley = KeithleyDAQ6510("GPIB::1")
keithley = KeithleyDAQ6510("TCPIP::192.168.1.1::INSTR")
print(keithley.current)
                                       # Prints the current in Amps
keithley.current_range = 10E-6
                                       # Select the 10 uA range
print(keithley.voltage)
                                       # Prints the voltage in Volts
keithley.voltage_range = 100e-3
                                      # Select the 100 mV range
print(keithley.resistance)
                                       # Prints the resistance in Ohms
keithley.offset_compensated = "ON" # Turns offset-compensated ohms on
keithley.open_channels([134, 135])
                                       # Open channels 134 and 135 on the MUX card
keithley.close_channel(133)
                                       # Close channel 133 on the MUX card
```

beep(frequency, duration)

Sound a system beep.

### **Parameters**

- **frequency** A frequency in Hz from 20 and 8000 Hz
- **duration** The amount of time to play the tone, between 0.001 s to 100 s

### Returns

None

### property buffer\_data

Get a numpy array of values from the buffer.

# property buffer\_points

Control the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

### check errors()

Read all errors from the instrument.

### Returns

List of error entries.

# check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

### clear()

Clear the instrument status byte.

# close\_channel(channel)

Set a single channel to closed.

### **Parameters**

**channel** – Channel to be set to closed.

### close\_channels(channel list)

Configure multiple channels to be closed.

#### **Parameters**

**channel\_list** – List of channels to be set to closed.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

### config\_buffer(points=64, delay=0)

Configure the measurement buffer for a number of points, to be taken with a specified delay.

### **Parameters**

- **points** The number of points in the buffer.
- **delay** The delay time in seconds.

# property current

Measure the current in Amps, if configured for this reading.

# property current\_nplc

Control the number of power line cycles (NPLC) for the DC current measurements, which sets the integration period and measurement speed. Takes values from 5E-4 to 15 (60 Hz) or 12 (50 Hz or 400 Hz). The smallest value is the shortest time, and results in the fastest reading rate, but increases the reading noise and decreases the number of usable digits. The largest value is the longest time, and results in the lowest reading rate, but increases the number of usable digits.

### property current\_range

Control the measurement current range in Amps. If the measurement function is DC, available ranges are 10E-6 A to 3A. If the measurement function is AC, available ranges are 100E-6 to 3A. Auto-range is disabled when this property is set.

# disable\_buffer()

Disable the connection between measurements and the buffer, but does not abort the measurement process.

### property id

Get the identification of the instrument.

# is\_buffer\_full()

Return True if the buffer is full of measurements.

# measure\_current(nplc=1, current=3, auto\_range=True)

Configure the measurement of current.

### **Parameters**

- nplc Number of power line cycles (NPLC) from 5E-4 to 15 (60 Hz) or 12 (50 Hz or 400 Hz).
- current Upper limit of current in Amps, from 10E-6 to 3 A (DC) or 100E-6 to 3A (AC).
- auto\_range A boolean value to enable auto\_range if True, else uses the set current.

**measure\_resistance**(*nplc=1*, *resistance=100000000.0*, *auto\_range=True*)

Configure the measurement of resistance.

### **Parameters**

- nplc Number of power line cycles (NPLC) from 5E-4 to 15 (60 Hz) or 12 (50 Hz or 400 Hz).
- **resistance** Upper limit of resistance in Ohms, from 10 to 100E6 (2-wire), 1 to 100E6 (4-wire with OCOM off), or 1 to 10E3 (4-wire with OCOM on).
- auto\_range A boolean value to enable auto\_range if True, else uses the set resistance.

measure\_voltage(nplc=1, voltage=1000, auto\_range=True)

Configure the measurement of voltage.

### **Parameters**

- **nplc** Number of power line cycles (NPLC) from 5E-4 to 15 (60 Hz) or 12 (50 Hz or 400 Hz).
- voltage Upper limit of voltage in Volts, from 100E-3 to 1000 V (DC) or 100E-3 to 750 V (AC).
- auto\_range A boolean value to enable auto\_range if True, else uses the set voltage.

### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# property offset\_compensated

Control if offset compensation is used or not. Valid values are OFF, ON, and AUTO.

### open\_channel(channel)

Set a single channel to open.

# **Parameters**

**channel** – Channel to be set to open.

# open\_channels(channel\_list)

Configure multiple channels to be open.

### **Parameters**

**channel\_list** – List of channels to be set to open.

# property options

Get the device options installed.

### read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument.

### reset buffer()

Reset the buffer.

# property resistance

Measure the resistance in Ohms, if configured for this reading.

### property resistance\_nplc

Control the number of power line cycles (NPLC) for the 2-wire resistance measurements, which sets the integration period and measurement speed. Takes values from 5E-4 to 15 (60 Hz) or 12 (50 Hz or 400 Hz). The smallest value is the shortest time, and results in the fastest reading rate, but increases the reading noise and decreases the number of usable digits. The largest value is the longest time, and results in the lowest reading rate, but increases the number of usable digits.

### property resistance\_range

Control the resistance range in Ohms. If the measurement function is 2-wire, the available ranges are 10 to 100E6 Ohms. If the measurement function is 4-wire resistance with offset compensation off, the available ranges are 1 to 100E6 Ohms. If the measurement function is 4-wire resistance with offset compensation on, the available ranges are 1 to 10E3 Ohms. Auto-range is disabled when this property is set.

# property sense\_mode

Control the reading mode, which can take the values 'current' or 'voltage'. The convenience methods sense\_current() and sense\_voltage() can also be used.

### shutdown()

Brings the instrument to a safe and stable state

### start\_buffer()

Starts the buffer.

# property status

Get the status byte and Master Summary Status bit.

# stop\_buffer()

Abort the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

### property voltage

Measure the voltage in Volts, if configured for this reading.

# property voltage\_nplc

Control the number of power line cycles (NPLC) for the DC voltage measurements, which sets the integration period and measurement speed. Takes values from 5E-4 to 15 (60 Hz) or 12 (50 Hz or 400 Hz). The smallest value is the shortest time, and results in the fastest reading rate, but increases the reading noise and decreases the number of usable digits. The largest value is the longest time, and results in the lowest reading rate, but increases the number of usable digits.

### property voltage\_range

Control the measurement voltage range in Volts. If the measurement function is DC, available ranges are 100E-3 V to 1000 V. If the measurement function is AC, available ranges are 100E-3 to 750 V. Auto-range is disabled when this property is set.

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Block the program, waiting for a full buffer. This function returns early if the should\_stop function returns True or the timeout is reached before the buffer is full.

### **Parameters**

- should\_stop A function that returns True when this function should return early
- **timeout** A time in seconds after which this function should return early
- interval A time in seconds for how often to check if the buffer is full

# write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

# 7.31.17 Keithley 4200A-SCS Parameter Analyzer

class pymeasure.instruments.keithley.Keithley4200(adapter, name='Keithley 4200A-SCS', \*\*kwargs)

Bases: Instrument

A class representing the Keithley 4200A-SCS Parameter Analyzer.

This driver only uses the user mode commands for controlling the SMUs.

Start the 'KXCI' program on the 4200A-SCS to activate remote control. The remote interface is configured with the Keithley Configuration Utility 'KCon'.

Currently, the driver is only working with the ethernet interface.

### add\_smu(id)

Add a SMU channel to the device.

### check\_set\_errors()

Check for errors after sending a command.

#### Raise

ValueError if response is not 'ACK'

### clear()

Clear all data from the buffer.

It also clears bit B0 (DATA\_READY) of the status byte.

# property id

Get the identification of the instrument (str).

# property options

Get the installed options (list of str).

# property status

Get the status byte (IntFlag).

class pymeasure.instruments.keithley.keithley4200.SMU(parent, id)

Bases: Channel

A class representing the SMU (source/measure unit) channel.

# property current

Measure the current in Amps.

# property current\_setpoint

Set range, output current and voltage compliance (int, float, float).

Current is in Amps and voltage in Volts.

```
inst = Keithley4200("TCPIP::192.168.1.1::1225::SOCKET")
inst.smu1.current_setpoint = (9, 55e-3, 10) # (range, value, compliance)
# Set SMU1 to output 55 mA in 100 mA range with 10 V voltage compliance
```

# **Current source range:**

- 0: autorange
- 1: 1 nA range, only with a preamplifier
- 2: 10 nA range, only with a preamplifier

- 3: 100 nA range
- 4: 1 µA range
- 5: 10 µA range
- 6: 100 µA range
- 7: 1 mA range
- 8: 10 mA range
- 9: 100 mA range
- 10: 1 A range, only with a 4210-SMU or 4211-SMU
- 11: 1 pA range, only with a preamplifier
- 12: 10 pA range, only with a preamplifier
- 13: 100 pA range, only with a preamplifier

### disable()

Disable the SMU.

# property voltage

Measure the voltage in Volts (float).

# property voltage\_setpoint

Set range, output voltage and current compliance (int, float, float).

Voltage is in Volts and current in Amps.

```
inst = Keithley4200("TCPIP::192.168.1.1::1225::SOCKET")
inst.smu1.voltage_setpoint = (0, 3.3, 1e-2) # (range, value, compliance)
# Set SMU1 to output 3.3 V in autorange with 10 mA current compliance
```

# Voltage source range:

- 0: autorange
- 1: 20 V range
- 2: 200 V range
- 3: 200 V range
- 4: 200 mV range, only with a preamplifier
- 5: 2 V range, only with a preamplifier

Bases: IntFlag

A class representing the status codes ofd the Keithley 4200.

### **Status codes:**

- 0: NONE
- 1: DATA\_READY

2: SYNTAX ERROR

• 16: BUSY

• 64: SERVICE\_REQUEST

# 7.32 KEPCO INC.

This section contains specific documentation on the Kepco Inc. power supplies instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.32.1 BOP Bipolar Power Supply with BIT 4886 Digital Interface Card

Bases: SCPIMixin, Instrument

Represents the Kepco BOP 36-12 (M or D) 400 W bipolar power supply fitted with BIT 4886 digital interface card (minimal implementation) and provides a high-level interface for interacting with the instrument.

### beep()

Cause the unit to emit a brief audible tone.

# property bop\_test

Get error code after performing full power supply self-test.

Returns 0 if all tests passed, otherwise corresponding error code as detailed in manual. Caution: Output will switch on and swing to maximum values. Disconnect any load before testing.

# property confidence\_test

Get error code after performing interface self-test procedure.

Returns 0 if all tests passed, otherwise corresponding error code as detailed in manual.

### property current

Measure current through the output terminals in Amps.

# property current\_setpoint

Control the output current setpoint.

Functionality depends on the operating mode. If power supply in current mode, this sets the output current setpoint. The current achieved depends on the voltage compliance and load conditions (see: *current*). If power supply in voltage mode, this sets the compliance current for the corresponding voltage set point. Query returns programmed value, meaning of which is dependent on power supply operating context (see: *operating\_mode*).

Output must be enabled separately (see: output\_enabled)

# property operating\_mode

Control the operating mode of the BOP.

As a command, a string, VOLT or CURR, is sent. As a query, a 0 or 1 is returned, corresponding to VOLT or CURR respectively. This is mapped to corresponding string.

# property output\_enabled

Control whether the source is enabled, takes values True or False (bool)

7.32. KEPCO INC. 407

### property voltage

Measure voltage present across the output terminals in Volts.

# property voltage\_setpoint

Control the output voltage setpoint.

Functionality depends on the operating mode. If power supply in voltage mode, this sets the output voltage setpoint. The voltage achieved depends on the current compliance and load conditions (see: *voltage*). If power supply in current mode, this sets the compliance voltage for the corresponding current set point. Query returns programmed value, meaning of which is dependent on power supply operating context (see: *operating\_mode*).

Output must be enabled separately (see: output\_enabled)

### wait\_to\_continue()

Cause the power supply to wait until all previously issued commands and queries are complete before executing subsequent commands or queries.

# 7.33 Keysight

This section contains specific documentation on the keysight instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

If the instrument you are looking for is not here, also check *Agilent* or *HP* for older instruments.

# 7.33.1 Keysight DSOX1102G Oscilloscope

Bases: SCPIUnknownMixin, Instrument

Represents the Keysight DSOX1102G Oscilloscope interface for interacting with the instrument.

Refer to the Keysight DSOX1102G Oscilloscope Programmer's Guide for further details about using the lower-level methods to interact directly with the scope.

```
scope = KeysightDSOX1102G(resource)
scope.autoscale()
ch1_data_array, ch1_preamble = scope.download_data(source="channel1", points=2000)
# ...
scope.shutdown()
```

### Known issues:

• The digitize command will be completed before the operation is. May lead to VI\_ERROR\_TMO (timeout) occurring when sending commands immediately after digitize. Current fix: if deemed necessary, add delay between digitize and follow-up command to scope.

# property acquisition\_mode

A string parameter that sets the acquisition mode. Can be "realtime" or "segmented".

### property acquisition\_type

A string parameter that sets the type of data acquisition. Can be "normal", "average", "hresolution", or "peak".

### autoscale()

Autoscale displayed channels.

# clear\_status()

Clear device status.

# default\_setup()

Default setup, some user settings (like preferences) remain unchanged.

# digitize(source: str)

Acquire waveforms according to the settings of the :ACQuire commands. Ensure a delay between the digitize operation and further commands, as timeout may be reached before digitize has completed. :param source: "channel1", "channel2", "function", "math", "fft", "abus", or "ext".

# download\_data(source, points=62500)

Get data from specified source of oscilloscope. Returned objects are a np.ndarray of data values (no temporal axis) and a dict of the waveform preamble, which can be used to build the corresponding time values for all data points.

Multimeter will be stopped for proper acquisition.

### **Parameters**

- **source** measurement source, can be "channel1", "channel2", "function", "fft", "wmemory1", "wmemory2", or "ext".
- **points** integer number of points to acquire. Note that oscilloscope may return fewer points than specified, this is not an issue of this library. Can be 100, 250, 500, 1000, 2000, 5000, 10000, 20000, 50000, or 62500.

# Return data\_ndarray, waveform\_preamble\_dict

see waveform\_preamble property for dict format.

# download\_image(format\_='png', color\_palette='color')

Get image of oscilloscope screen in bytearray of specified file format.

### **Parameters**

- format "bmp", "bmp8bit", or "png"
- color\_palette "color" or "grayscale"

# factory\_reset()

Factory default setup, no user settings remain unchanged.

# run()

Starts repetitive acquisitions.

This is the same as pressing the Run key on the front panel.

### single()

Causes the instrument to acquire a single trigger of data. This is the same as pressing the Single key on the front panel.

### stop()

Stops the acquisition. This is the same as pressing the Stop key on the front panel.

# property system\_setup

A string parameter that sets up the oscilloscope. Must be in IEEE 488.2 format. It is recommended to only set a string previously obtained from this command.

# property timebase

Read timebase setup as a dict containing the following keys: - "REF": position on screen of timebase reference (str) - "MAIN:RANG": full-scale timebase range (float) - "POS": interval between trigger and reference point (float) - "MODE": mode (str)

### property timebase\_mode

A string parameter that sets the current time base. Can be "main", "window", "xy", or "roll".

### property timebase\_offset

A float parameter that sets the time interval in seconds between the trigger event and the reference position (at center of screen by default).

# property timebase\_range

A float parameter that sets the full-scale horizontal time in seconds for the main window.

### property timebase\_scale

A float parameter that sets the horizontal scale (units per division) in seconds for the main window.

# timebase\_setup(mode=None, offset=None, horizontal\_range=None, scale=None)

Set up timebase. Unspecified parameters are not modified. Modifying a single parameter might impact other parameters. Refer to oscilloscope documentation and make multiple consecutive calls to channel\_setup if needed.

#### **Parameters**

- mode Timebase mode, can be "main", "window", "xy", or "roll".
- offset Offset in seconds between trigger and center of screen.
- horizontal\_range Full-scale range in seconds.
- **scale** Units-per-division in seconds.

# property waveform\_data

Get the binary block of sampled data points transmitted using the IEEE 488.2 arbitrary block data format.

### property waveform\_format

A string parameter that controls how the data is formatted when sent from the oscilloscope. Can be "ascii", "word" or "byte". Words are transmitted in big endian by default.

# property waveform\_points

An integer parameter that sets the number of waveform points to be transferred with the waveform\_data method. Can be any of the following values: 100, 250, 500, 1000, 2 000, 5 000, 10 000, 20 000, 50 000, 62 500.

Note that the oscilloscope may provide less than the specified nb of points.

# property waveform\_points\_mode

A string parameter that sets the data record to be transferred with the waveform\_data method. Can be "normal", "maximum", or "raw".

### property waveform\_preamble

Get preamble information for the selected waveform source as a dict with the following keys: - "format": byte, word, or ascii (str) - "type": normal, peak detect, or average (str) - "points": nb of data points transferred (int) - "count": always 1 (int) - "xincrement": time difference between data points (float) - "xorigin": first data point in memory (float) - "xreference": data point associated with xorigin (int) - "yincrement": voltage difference between data points (float) - "yorigin": voltage at center of screen (float) - "yreference": data point associated with yorigin (int)

### property waveform\_source

A string parameter that selects the analog channel, function, or reference waveform to be used as the source for the waveform methods. Can be "channel1", "channel2", "function", "fft", "wmemory1", "wmemory2", or "ext".

# 7.33.2 Keysight N5767A Power Supply

Bases: SCPIUnknownMixin, Instrument

Represents the Keysight N5767A Power supply interface for interacting with the instrument.

# property current

Get current in Amps.

### property current\_range

Control the DC current range in Amps, which can take values from 0 to 25 A. Auto-range is disabled when this property is set. (float)

# disable()

Disables the flow of current.

### enable()

Enables the flow of current.

### is\_enabled()

Returns True if the current supply is enabled.

# property voltage

Get a DC voltage measurement in Volts.

# property voltage\_range

Control the DC voltage range in Volts, which can take values from 0 to 60 V. Auto-range is disabled when this property is set.

# 7.33.3 Keysight N5776C Power Supply

Bases: SCPIUnknownMixin, Instrument

This represents the Keysight N7776C Tunable Laser Source interface.

```
laser = N7776C(address)
laser.sweep_wl_start = 1550
laser.sweep_wl_stop = 1560
laser.sweep_speed = 1
laser.sweep_mode = 'CONT'
laser.output_enabled = 1
while laser.sweep_state == 1:
    log.info('Sweep in progress.')
laser.output_enabled = 0
```

### close()

Fully closes the connection to the instrument through the adapter connection.

# get\_wl\_data()

Function returning the wavelength data logged in the internal memory of the laser

### property locked

Control the lock state (True/False) of the laser source. (bool)

### next\_step()

Performs the next sweep step in stepped sweep if it is paused or in manual mode.

### property output\_enabled

Control the state (on/off) of the laser source (bool)

### property output\_power\_dBm

Control the output power in dBm.

### property output\_power\_mW

Control the output power in mW

# previous\_step()

Performs one sweep step backwards in stepped sweep if its paused or in manual mode.

# property sweep\_mode

Control sweep mode of the swept laser source

### property sweep\_points

Get the number of datapoints that the :READout:DATA? command will return.

# property sweep\_speed

Control speed of the sweep (in nanometers per second).

# property sweep\_state

Control state of the wavelength sweep. Stops, starts, pauses or continues a wavelength sweep. Possible state values are 0 (not running), 1 (running) and 2 (paused). Refer to the N7776C user manual for exact usage of the paused option.

# property sweep\_step

Control step width of the sweep (in nanometers).

### property sweep\_twoway

Control the repeat mode. Applies in stepped, continuous and manual sweep mode.

# property sweep\_wl\_start

Control Start Wavelength (in nanometers) for a sweep.

# property sweep\_wl\_stop

Control End Wavelength (in nanometers) for a sweep.

### property trigger\_in

Control the incoming trigger response and arm the module.

# property trigger\_out

Control if and at which point in a sweep cycle an output trigger is generated and arms the module.

# property wavelength

Control absolute wavelength of the output light (in nanometers)

# property wl\_logging

Control State (on/off) of the lambda logging feature of the laser source.

# 7.33.4 Keysight E36312A Triple Output Power Supply

class pymeasure.instruments.keysight.KeysightE36312A(adapter, name='Keysight E36312A', \*\*kwargs)

Bases: SCPIMixin, Instrument

Represents the Keysight E36312A Power supply interface for interacting with the instrument.

```
supply = KeysightE36312A(resource)
supply.ch_1.voltage_setpoint=10
supply.ch_1.current_setpoint=0.1
supply.ch_1.output_enabled=True
print(supply.ch_1.voltage)
```

# $ch_1$

### Channel

VoltageChannel

ch\_2

### Channel

VoltageChannel

 $ch_3$ 

# Channel

VoltageChannel

# class BaseChannelCreator(cls, \*\*kwargs)

Bases: object

Base class for ChannelCreator and MultiChannelCreator.

# **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- \*\*kwargs Keyword arguments for all children.

### class ChannelCreator(cls, id=None, \*\*kwargs)

Bases: BaseChannelCreator

Add a single channel to the parent class.

The child will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name that ChannelCreator was assigned to in the Instrument class will be the name of the channel interface.

```
class Extreme5000(Instrument):
   # Two output channels, accessible by their property names
   # and both are accessible through the 'channels' collection
   output_A = Instrument.ChannelCreator(Extreme5000Channel, "A")
   output_B = Instrument.ChannelCreator(Extreme5000Channel, "B")
   # A channel without a channel accessible through the 'motor' collection
```

7.33. Keysight 413

(continues on next page)

(continued from previous page)

```
motor = Instrument.ChannelCreator(MotorControl)
inst = SomeInstrument()
# Set the extreme_temp for channel A of Extreme5000 instrument
inst.output_A.extreme_temp = 42
```

#### **Parameters**

- cls Channel class for channel interface
- id The id of the channel on the instrument, integer or string.
- \*\*kwargs Keyword arguments for all children.

class MultiChannelCreator(cls, id=None, prefix='ch\_', \*\*kwargs)

Bases: BaseChannelCreator

Add channels to the parent class.

The children will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name (e.g. channels) will be used as the *collection* of the children. You may define the attribute prefix. If there are no other pressing reasons, use channels as the attribute name and leave the prefix at the default "ch\_".

### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- id tuple/list of ids of the channels on the instrument.
- **prefix** Collection prefix for the attributes, e.g. "*ch*\_" creates attribute *self.ch*\_A. If prefix evaluates False, the child will be added directly under the variable name. Required if id is tuple/list.
- **\*\*kwargs** Keyword arguments for all children.

add\_child(cls, id=None, collection='channels', prefix='ch\_', attr\_name='', \*\*kwargs)

Add a child to this instance and return its index in the children list.

The newly created child may be accessed either by the id in the children dictionary or by the created attribute, e.g. the fifth channel of *instrument* with id "F" has two access options: instrument.channels["F"] == instrument.ch\_F

### 1 Note

Do not change the default *collection* or *prefix* parameter, unless you have to distinguish several collections of different children, e.g. different channel types (analog and digital).

### **Parameters**

- cls Class of the channel.
- id Child id how it is used in communication, e.g. "A".
- collection Name of the collection of children, used for dictionary access to the channel interfaces.
- prefix For creating multiple channel interfaces, the prefix e.g. "ch\_" is prepended to the attribute name of the channel interface self.ch\_A. If prefix evaluates False, the child will be added directly under the collection name.
- attr\_name For creating a single channel interface, the attr\_name argument is used when setting the attribute name of the channel interface.
- **\*\*kwargs** Keyword arguments for the channel creator.

### Returns

Instance of the created child.

# ask(command, query\_delay=None)

Write a command to the instrument and return the read response.

### **Parameters**

- **command** Command string to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.

### Returns

String returned by the device without read\_termination.

# binary\_values(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return a numpy array of the binary data.

# **Parameters**

- **command** Command to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- **kwargs** Arguments for read\_binary\_values().

# **Returns**

NumPy array of values.

# check errors()

Read all errors from the instrument.

### Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

# clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

Return a property for the class based on the supplied commands. This property may be set and read from the instrument. See also *measurement()* and *setting()*.

### **Parameters**

- **get\_command** A string command that asks for the value, set to *None* if get is not supported (see also *setting()*).
- **set\_command** A string command that writes the value, set to *None* if set is not supported (see also *measurement()*).
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- get\_process A function that takes a value and allows processing before value mapping, returning the processed value
- get\_process\_list A function that takes the value list and processes it.

- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that takes a command and allows processing before executing the command

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_set\_errors** Toggles checking errors after setting
- check\_get\_errors Toggles checking errors after getting
- dynamic Specify whether the property parameters are meant to be changed in instances or subclasses.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- values\_kwargs (dict) Further keyword arguments for values().
- \*\*kwargs Keyword arguments for values().

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

Example of usage of dynamic parameter is as follows:

```
class GenericInstrument(Instrument):
    center_frequency = Instrument.control(
        ":SENS:FREQ:CENT?;", ":SENS:FREQ:CENT %e GHz;",
        " A floating point property that represents the frequency ... ",
        validator=strict_range,
        # Redefine this in subclasses to reflect actual instrument value:
        values=(1, 20),
        dynamic=True # enable changing property parameters on-the-fly
)

class SpecificInstrument(GenericInstrument):
    # Identical to GenericInstrument, except for frequency range
    # Override the "values" parameter of the "center_frequency" property
    center_frequency_values = (1, 10) # Redefined at subclass level

instrument = SpecificInstrument()
instrument.center_frequency_values = (1, 6e9) # Redefined at instance level
```

# **Marning**

Unexpected side effects when using dynamic properties

Users must pay attention when using dynamic properties, since definition of class and/or instance attributes matching specific patterns could have unwanted side effect. The attribute name pattern *property\_param*, where *property* is the name of the dynamic property (e.g. *center\_frequency* in the example) and *param* 

is any of this method parameters name except *dynamic* and *docs* (e.g. *values* in the example) has to be considered reserved for dynamic property control.

### static get\_channel\_pairs(cls)

Return a list of all the Instrument's channel pairs

### static get\_channels(cls)

Return a list of all the Instrument's ChannelCreator and MultiChannelCreator instances

### property id

Get the identification of the instrument.

static measurement(get\_command, docs, values=(), map\_values=False, get\_process=<function

CommonBase.<lambda>>, get\_process\_list=<function CommonBase.<lambda>>,

command\_process=None, check\_get\_errors=False, dynamic=False,

preprocess\_reply=None, separator=', ', maxsplit=-1, cast=<class 'float'>,

values\_kwargs=None, \*\*kwargs)

Return a property for the class based on the supplied commands. This is a measurement quantity that may only be read from the instrument, not set.

#### **Parameters**

- **get\_command** A string command that asks for the value
- docs A docstring that will be included in the documentation
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **command\_process** A function that take a command and allows processing before executing the command, for getting

Deprecated since version 0.12: Use a dynamic property instead.

- check\_get\_errors Toggles checking errors after getting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- **values\_kwargs** (*dict*) Further keyword arguments for *values*().
- \*\*kwargs Keyword arguments for *values()*.

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

### property options

Get the device options installed.

```
read(**kwargs)
```

Read up to (excluding) read\_termination or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

```
read_bytes(count, **kwargs)
```

Read a certain number of bytes from the instrument.

### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

### remove\_child(child)

Remove the child from the instrument and the corresponding collection.

#### **Parameters**

**child** – Instance of the child to delete.

### reset()

Reset the instrument.

Return a property for the class based on the supplied commands. This property may be set, but raises an exception when being read from the instrument.

### **Parameters**

- set\_command A string command that writes the value
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **check\_set\_errors** Toggles checking errors after setting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.

# shutdown()

Brings the instrument to a safe and stable state

### property status

Get the status byte and Master Summary Status bit.

**values**(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None, maxsplit=-1, \*\*kwargs)

Write a command to the instrument and return a list of formatted values from the result.

### **Parameters**

- **command** SCPI command to be sent to the instrument.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- \*\*kwargs Keyword arguments to be passed to the ask() method.

#### Returns

A list of the desired type, or strings where the casting fails.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### Parameters

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

# **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

class pymeasure.instruments.keysight.keysightE36312A.VoltageChannel(parent, id)

Bases: Channel

### property current

Measure the actual current of this channel.

# property current\_limit

Control the current limit of this channel, range depends on channel.(dynamic)

### property output\_enabled

Control whether the channel output is enabled (boolean).

# property voltage

Measure actual voltage of this channel.

# property voltage\_setpoint

Control the output voltage of this channel, range depends on channel.(dynamic)

# 7.33.5 Keysight E3631A Triple Output Power Supply

class pymeasure.instruments.keysight.KeysightE3631A(adapter, name='Keysight E3631A', \*\*kwargs)

Bases: SCPIMixin, Instrument

Represents the Keysight E3631A Triple Output DC Power Supply interface for interacting with the instrument.

```
supply = KeysightE3631A(resource)
supply.ch_1.voltage_setpoint=10
supply.ch_1.current_setpoint=0.1
supply.ch_1.output_enabled=True
print(supply.ch_1.voltage)
```

# ch\_1

### Channel

VoltageChannel

 $ch_2$ 

# Channel

VoltageChannel

 $ch_3$ 

### Channel

VoltageChannel

# class BaseChannelCreator(cls, \*\*kwargs)

Bases: object

Base class for ChannelCreator and MultiChannelCreator.

### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- \*\*kwargs Keyword arguments for all children.

class ChannelCreator(cls, id=None, \*\*kwargs)

Bases: BaseChannelCreator

Add a single channel to the parent class.

The child will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name that ChannelCreator was assigned to in the *Instrument* class will be the name of the channel interface.

```
class Extreme5000(Instrument):
    # Two output channels, accessible by their property names
    # and both are accessible through the 'channels' collection
    output_A = Instrument.ChannelCreator(Extreme5000Channel, "A")
    output_B = Instrument.ChannelCreator(Extreme5000Channel, "B")
    # A channel without a channel accessible through the 'motor' collection
    motor = Instrument.ChannelCreator(MotorControl)

inst = SomeInstrument()
# Set the extreme_temp for channel A of Extreme5000 instrument
inst.output_A.extreme_temp = 42
```

#### **Parameters**

- cls Channel class for channel interface
- id The id of the channel on the instrument, integer or string.
- **\*\*kwargs** Keyword arguments for all children.

```
class MultiChannelCreator(cls, id=None, prefix='ch_', **kwargs)
```

Bases: BaseChannelCreator

Add channels to the parent class.

The children will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name (e.g. channels) will be used as the *collection* of the children. You may define the attribute prefix. If there are no other pressing reasons, use channels as the attribute name and leave the prefix at the default "ch\_".

# **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- id tuple/list of ids of the channels on the instrument.
- **prefix** Collection prefix for the attributes, e.g. "*ch*\_" creates attribute *self.ch*\_A. If prefix evaluates False, the child will be added directly under the variable name. Required if id is tuple/list.
- \*\*kwargs Keyword arguments for all children.

add\_child(cls, id=None, collection='channels', prefix='ch\_', attr\_name='', \*\*kwargs)

Add a child to this instance and return its index in the children list.

The newly created child may be accessed either by the id in the children dictionary or by the created attribute, e.g. the fifth channel of *instrument* with id "F" has two access options: instrument.channels["F"] == instrument.ch\_F

### 1 Note

Do not change the default *collection* or *prefix* parameter, unless you have to distinguish several collections of different children, e.g. different channel types (analog and digital).

### **Parameters**

- **cls** Class of the channel.
- id Child id how it is used in communication, e.g. "A".
- collection Name of the collection of children, used for dictionary access to the channel interfaces.
- prefix For creating multiple channel interfaces, the prefix e.g. "ch\_" is prepended to the attribute name of the channel interface self.ch\_A. If prefix evaluates False, the child will be added directly under the collection name.
- attr\_name For creating a single channel interface, the attr\_name argument is used when setting the attribute name of the channel interface.
- **\*\*kwargs** Keyword arguments for the channel creator.

### Returns

Instance of the created child.

### **ask**(command, query delay=None)

Write a command to the instrument and return the read response.

# **Parameters**

- **command** Command string to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.

# **Returns**

String returned by the device without read\_termination.

# binary\_values(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return a numpy array of the binary data.

### **Parameters**

- **command** Command to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- **kwargs** Arguments for read\_binary\_values().

### Returns

NumPy array of values.

### check\_errors()

Read all errors from the instrument.

# Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

### Returns

List of error entries.

# clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

```
static control (get_command, set_command, docs, validator=<function CommonBase.<lambda>>, values=(), map_values=False, get_process=<function CommonBase.<lambda>>, get_process_list=<function CommonBase.<lambda>>, set_process=<function CommonBase.<lambda>>, command_process=None, check_set_errors=False, check_get_errors=False, dynamic=False, preprocess_reply=None, separator=', ', maxsplit=-1, cast=<class 'float'>, values_kwargs=None, **kwargs')
```

Return a property for the class based on the supplied commands. This property may be set and read from the instrument. See also *measurement()* and *setting()*.

### **Parameters**

- **get\_command** A string command that asks for the value, set to *None* if get is not supported (see also *setting()*).
- **set\_command** A string command that writes the value, set to *None* if set is not supported (see also *measurement()*).
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- get\_process A function that takes a value and allows processing before value mapping, returning the processed value
- get\_process\_list A function that takes the value list and processes it.

- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that takes a command and allows processing before executing the command

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_set\_errors** Toggles checking errors after setting
- check\_get\_errors Toggles checking errors after getting
- dynamic Specify whether the property parameters are meant to be changed in instances or subclasses.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- values\_kwargs (dict) Further keyword arguments for values().
- \*\*kwargs Keyword arguments for values().

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

Example of usage of dynamic parameter is as follows:

```
class GenericInstrument(Instrument):
    center_frequency = Instrument.control(
        ":SENS:FREQ:CENT?;", ":SENS:FREQ:CENT %e GHz;",
        " A floating point property that represents the frequency ... ",
        validator=strict_range,
        # Redefine this in subclasses to reflect actual instrument value:
        values=(1, 20),
        dynamic=True # enable changing property parameters on-the-fly
)

class SpecificInstrument(GenericInstrument):
    # Identical to GenericInstrument, except for frequency range
    # Override the "values" parameter of the "center_frequency" property
    center_frequency_values = (1, 10) # Redefined at subclass level

instrument = SpecificInstrument()
instrument.center_frequency_values = (1, 6e9) # Redefined at instance level
```

# **Marning**

Unexpected side effects when using dynamic properties

Users must pay attention when using dynamic properties, since definition of class and/or instance attributes matching specific patterns could have unwanted side effect. The attribute name pattern *property\_param*, where *property* is the name of the dynamic property (e.g. *center\_frequency* in the example) and *param* 

is any of this method parameters name except *dynamic* and *docs* (e.g. *values* in the example) has to be considered reserved for dynamic property control.

### static get\_channel\_pairs(cls)

Return a list of all the Instrument's channel pairs

### static get\_channels(cls)

Return a list of all the Instrument's ChannelCreator and MultiChannelCreator instances

### property id

Get the identification of the instrument.

Return a property for the class based on the supplied commands. This is a measurement quantity that may only be read from the instrument, not set.

#### **Parameters**

- **get\_command** A string command that asks for the value
- docs A docstring that will be included in the documentation
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **command\_process** A function that take a command and allows processing before executing the command, for getting

Deprecated since version 0.12: Use a dynamic property instead.

- check\_get\_errors Toggles checking errors after getting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- **maxsplit** The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- **values\_kwargs** (*dict*) Further keyword arguments for *values*().
- \*\*kwargs Keyword arguments for *values()*.

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

#### property options

Get the device options installed.

# property output\_enabled

Control whether the output of the last used channel is enabled (boolean).(dynamic)

```
read(**kwargs)
```

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

```
read_bytes(count, **kwargs)
```

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### remove\_child(child)

Remove the child from the instrument and the corresponding collection.

#### **Parameters**

**child** – Instance of the child to delete.

# reset()

Reset the instrument.

Return a property for the class based on the supplied commands. This property may be set, but raises an exception when being read from the instrument.

#### **Parameters**

- **set\_command** A string command that writes the value
- **docs** A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **check\_set\_errors** Toggles checking errors after setting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.

#### shutdown()

Brings the instrument to a safe and stable state

### property status

Get the status byte and Master Summary Status bit.

# property tracking\_enabled

Control whether the power supply operates in the track mode (boolean)

 $\textbf{values} (\textit{command}, \textit{separator} = ', ', \textit{cast} = < \textit{class'float'} >, \textit{preprocess\_reply} = \textit{None}, \textit{maxsplit} = -1, **kwargs)$ 

Write a command to the instrument and return a list of formatted values from the result.

#### **Parameters**

- **command** SCPI command to be sent to the instrument.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- \*\*kwargs Keyword arguments to be passed to the ask() method.

### Returns

A list of the desired type, or strings where the casting fails.

```
wait_for(query_delay=None)
```

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

# **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

class pymeasure.instruments.keysight.keysightE3631A.VoltageChannel(parent, id)

Bases: Channel

Implementation of a power supply base class channel

#### property current

Measure the actual current of this channel.

### property current\_limit

Control the current limit of this channel, range depends on channel.(dynamic)

#### property output\_enabled

Control whether the channel output is enabled (boolean).

#### property voltage

Measure actual voltage of this channel.

# property voltage\_setpoint

Control the output voltage of this channel, range depends on channel.(dynamic)

# 7.33.6 Keysight 81160A Pulse Function Arbitrary Noise Generator

Bases: Agilent33500

Represent the Keysight 81160A and provide a high-level interface for interacting with the instrument.

```
generator = Keysight81160A("GPIB::1")
                                             # Replace with your device address
generator.reset()
                                             # Reset the generator to default.
→ settings
generator.shape = "SIN"
                                             # Set default channel output shape to.
⇔sine
generator.channels[1].shape = "SIN"
                                             # Set channel 1 output signal shape to.
⇔sine
generator.frequency = 1e3
                                            # Set default channel output frequency_
\rightarrowto 1 kHz
generator.channels[1].frequency = 1e3
                                            # Set channel 1 output frequency to 1.
generator.channels[2].amplitude = 5
                                             # Set channel 2 output amplitude to 5.
generator.channels[2].offset = 0.5
                                           # Set channel 2 output offset to 0.5 V
generator.channels[2].output = True
                                           # Enable channel 2 output
# Output a square wave at 1 kHz with 5 Vpp and 0.5 V offset
generator.channels[2].apply_square(1e3, 5, 0)
ch1 = generator.channels[1]
                                           # Short form for channel 1
waveform = [-2.0, -1.5, 0.0, 1.5, 2.0] # Define a user-defined waveform in_
\hookrightarrow Volts
ch1.waveform_volatile = waveform
                                             # Set user-defined waveform to_
→volatile memory
print(f"{ch1.free_memory_slots} slots free") # Get number of slots in non-volatile_
→memorv
ch1.save_waveform(waveform, "test")
                                             # Save the waveform to non-volatile_
\hookrightarrowmemory
```

(continues on next page)

(continued from previous page)

```
ch1.shape = "USER"  # Set channel 1 shape to user-defined
ch1.trigger_mode = "MAN"  # Set trigger mode to manual
ch1.trigger_count = 2  # Set number of cycles to be generated.

→ to 2
ch1.output = True  # Enable channel 1 output
generator.trigger()  # Trigger the generator to output the.

→ waveform
ch1.delete_waveform("test")  # Delete the waveform from non-
→ volatile memory
```

#### ch\_1

#### Channel

Keysight81160AChannel

ch\_2

#### Channel

Keysight81160AChannel

# class BaseChannelCreator(cls, \*\*kwargs)

Bases: object

Base class for ChannelCreator and MultiChannelCreator.

#### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- **\*\*kwargs** Keyword arguments for all children.

# class ChannelCreator(cls, id=None, \*\*kwargs)

Bases: BaseChannelCreator

Add a single channel to the parent class.

The child will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name that ChannelCreator was assigned to in the *Instrument* class will be the name of the channel interface.

```
class Extreme5000(Instrument):
    # Two output channels, accessible by their property names
    # and both are accessible through the 'channels' collection
    output_A = Instrument.ChannelCreator(Extreme5000Channel, "A")
    output_B = Instrument.ChannelCreator(Extreme5000Channel, "B")
    # A channel without a channel accessible through the 'motor' collection
    motor = Instrument.ChannelCreator(MotorControl)

inst = SomeInstrument()
# Set the extreme_temp for channel A of Extreme5000 instrument
inst.output_A.extreme_temp = 42
```

#### **Parameters**

- cls Channel class for channel interface
- id The id of the channel on the instrument, integer or string.
- **\*\*kwargs** Keyword arguments for all children.

class MultiChannelCreator(cls, id=None, prefix='ch\_', \*\*kwargs)

Bases: BaseChannelCreator

Add channels to the parent class.

The children will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name (e.g. channels) will be used as the collection of the children. You may define the attribute prefix. If there are no other pressing reasons, use channels as the attribute name and leave the prefix at the default "ch\_".

```
class Extreme5000(Instrument):
   # Three channels of the same type: 'ch_A', 'ch_B', 'ch_C'
   # and add them to the 'channels' collection
   channels = Instrument.MultiChannelCreator(Extreme5000Channel, ["A", "B", "C
"])
   # Two channel interfaces of different types: 'fn_power', 'fn_voltage'
   # and add them to the 'functions' collection
   functions = Instrument.MultiChannelCreator((PowerChannel, VoltageChannel),
                                     ["power", "voltage"], prefix="fn_")
```

#### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- id tuple/list of ids of the channels on the instrument.
- prefix Collection prefix for the attributes, e.g. "ch" creates attribute self.ch A. If prefix evaluates False, the child will be added directly under the variable name. Required if id is tuple/list.
- \*\*kwargs Keyword arguments for all children.

add\_child(cls, id=None, collection='channels', prefix='ch\_', attr\_name='', \*\*kwargs)

Add a child to this instance and return its index in the children list.

The newly created child may be accessed either by the id in the children dictionary or by the created attribute, e.g. the fifth channel of *instrument* with id "F" has two access options: instrument.channels["F"] == instrument.ch\_F



# 1 Note

Do not change the default *collection* or *prefix* parameter, unless you have to distinguish several collections of different children, e.g. different channel types (analog and digital).

#### **Parameters**

- **cls** Class of the channel.
- id Child id how it is used in communication, e.g. "A".
- collection Name of the collection of children, used for dictionary access to the channel interfaces.
- prefix For creating multiple channel interfaces, the prefix e.g. "ch\_" is prepended to the attribute name of the channel interface self.ch\_A. If prefix evaluates False, the child will be added directly under the collection name.

- attr\_name For creating a single channel interface, the attr\_name argument is used when setting the attribute name of the channel interface.
- **\*\*kwargs** Keyword arguments for the channel creator.

#### Returns

Instance of the created child.

# property amplitude

Control the voltage amplitude in Volts (float).(dynamic)

#### property amplitude\_unit

Control the amplitude units (string, strictly 'VPP', 'VRMS', or 'DBM').(dynamic)

#### property arb\_advance

Control how the device advances from data point to data point (str). Can be set to 'TRIG<GER>' or 'SRAT<E>' (default).

# property arb\_file

Control the arbitrary signal to use from the volatile memory of the device.(dynamic)

#### property arb\_filter

Control the filter setting for arbitrary signals (str). Can be set to 'NORM<AL>', 'STEP' and 'OFF'.

#### property arb\_srate

Control the sample rate of the currently selected arbitrary signal in Sa/s (float). Valid values range from 1  $\mu$ Sa/s to 250 MSa/s (maximum range, device-dependent).

# ask(command, query\_delay=None)

Write a command to the instrument and return the read response.

# **Parameters**

- **command** Command string to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.

### Returns

String returned by the device without read\_termination.

# beep()

Causes a system beep.

# binary\_values(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return a numpy array of the binary data.

#### **Parameters**

- **command** Command to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- **kwargs** Arguments for read\_binary\_values().

#### Returns

NumPy array of values.

### property burst\_mode

Control the burst mode type (str, strictly 'TRIG<GERED>' or 'GAT<ED>').(dynamic)

#### property burst\_ncycles

Control the number of cycles to be output when a burst is triggered (int).(dynamic)

# property burst\_period

Control the period of subsequent bursts in seconds (float).(dynamic)

#### property burst\_state

Control the burst mode state (bool).(dynamic)

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

# Returns

List of error entries.

#### clear()

Clear the instrument status byte.

#### clear\_display()

Removes a text message from the display.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

```
static control(get_command, set_command, docs, validator=<function CommonBase.<lambda>>, values=(), map_values=False, get_process=<function CommonBase.<lambda>>, get_process=list=<function CommonBase.<lambda>>, set_process=<function CommonBase.<lambda>>, command_process=None, check_set_errors=False, check_get_errors=False, dynamic=False, preprocess_reply=None, separator=', ', maxsplit=-1, cast=<class 'float'>, values_kwargs=None, **kwargs')
```

Return a property for the class based on the supplied commands. This property may be set and read from the instrument. See also *measurement()* and *setting()*.

#### **Parameters**

• **get\_command** – A string command that asks for the value, set to *None* if get is not supported (see also *setting()*).

- **set\_command** A string command that writes the value, set to *None* if set is not supported (see also *measurement()*).
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- get\_process A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that takes a command and allows processing before executing the command

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_set\_errors** Toggles checking errors after setting
- **check\_get\_errors** Toggles checking errors after getting
- dynamic Specify whether the property parameters are meant to be changed in instances or subclasses.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most maxsplit times. -1 (default) indicates no limit.
- **cast** A type to cast each element of the splitted string.
- **values\_kwargs** (*dict*) Further keyword arguments for *values*().
- \*\*kwargs Keyword arguments for values().

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

Example of usage of dynamic parameter is as follows:

```
class GenericInstrument(Instrument):
    center_frequency = Instrument.control(
        ":SENS:FREQ:CENT?;", ":SENS:FREQ:CENT %e GHz;",
        " A floating point property that represents the frequency ... ",
        validator=strict_range,
        # Redefine this in subclasses to reflect actual instrument value:
        values=(1, 20),
        dynamic=True # enable changing property parameters on-the-fly
)

class SpecificInstrument(GenericInstrument):
    # Identical to GenericInstrument, except for frequency range
```

(continues on next page)

(continued from previous page)

```
# Override the "values" parameter of the "center_frequency" property
   center_frequency_values = (1, 10) # Redefined at subclass level
instrument = SpecificInstrument()
instrument.center_frequency_values = (1, 6e9) # Redefined at instance level
```

#### Warning

Unexpected side effects when using dynamic properties

Users must pay attention when using dynamic properties, since definition of class and/or instance attributes matching specific patterns could have unwanted side effect. The attribute name pattern property\_param, where property is the name of the dynamic property (e.g. center\_frequency in the example) and param is any of this method parameters name except dynamic and docs (e.g. values in the example) has to be considered reserved for dynamic property control.

# **data\_arb**(arb\_name, data\_points, data\_format='DAC')

Uploads an arbitrary trace into the volatile memory of the device.

The data\_points can be given as: comma separated 16 bit DAC values (ranging from -32767 to +32767), as comma separated floating point values (ranging from -1.0 to +1.0) or as a binary data stream. Check the manual for more information. The storage depends on the device type and ranges from 8 Sa to 16 MSa (maximum).

#### **Parameters**

- arb\_name The name of the trace in the volatile memory. This is used to access the trace.
- data\_points Individual points of the trace. The format depends on the format parameter. format = 'DAC' (default): Accepts list of integer values ranging from -32767 to +32767. Minimum of 8 a maximum of 65536 points. format = 'float': Accepts list of floating point values ranging from -1.0 to +1.0. Minimum of 8 a maximum of 65536 points. format = 'binary': Accepts a binary stream of 8 bit data.
- data\_format Defines the format of data\_points. Can be 'DAC' (default), 'float' or 'binary'. See documentation on parameter data points above.

# data\_volatile\_clear()

Clear all arbitrary signals from volatile memory.

This should be done if the same name is used continuously to load different arbitrary signals into the memory, since an error will occur if a trace is loaded which already exists in the memory.

#### property display

Set text to be displayed on the front panel of the device (string).

#### property ext\_trig\_out

Control whether the trigger out signal is active (bool).

### property frequency

Control the waveform frequency in Hz (float). Depends on the specified shape.(dynamic)

# static get\_channel\_pairs(cls)

Return a list of all the Instrument's channel pairs

#### static get\_channels(cls)

Return a list of all the Instrument's ChannelCreator and MultiChannelCreator instances

# property id

Get the identification of the instrument.

static measurement(get\_command, docs, values=(), map\_values=False, get\_process=<function

CommonBase.<lambda>>, get\_process\_list=<function CommonBase.<lambda>>,

command\_process=None, check\_get\_errors=False, dynamic=False,

preprocess\_reply=None, separator=', ', maxsplit=-1, cast=<class 'float'>,

values\_kwargs=None, \*\*kwargs)

Return a property for the class based on the supplied commands. This is a measurement quantity that may only be read from the instrument, not set.

#### **Parameters**

- **get\_command** A string command that asks for the value
- docs A docstring that will be included in the documentation
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **command\_process** A function that take a command and allows processing before executing the command, for getting

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_get\_errors** Toggles checking errors after getting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- **maxsplit** The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- **values\_kwargs** (*dict*) Further keyword arguments for *values*().
- \*\*kwargs Keyword arguments for *values()*.

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

# property next\_error

Get the next error in the queue. If you want to read and log all errors, use <code>check\_errors()</code> instead.

#### property offset

Control the voltage offset in Volts (float).(dynamic)

#### property options

Get the device options installed.

# property output

Control the output state (bool).(dynamic)

# property output\_load

Control the expected load resistance in Ohms (str or float). The output impedance is always 50 Ohm, this setting can be used to correct the displayed voltage for loads unmatched to 50 Ohm.(dynamic)

#### property phase

Control the waveform phase in degrees (float, from -360 to 360).

#### phase\_sync()

Synchronize the phase of all channels.

# property pulse\_dutycycle

Control the pulse duty cycle in percent (float, from 0 to 100).(dynamic)

# property pulse\_hold

Control whether pulse width or duty cycle is maintained when the period or frequency of the waveform is changed (str).(dynamic)

# property pulse\_period

Control the pulse period in seconds (float). Overwrites frequency. If the period is shorter than the pulse width + the edge time, the edge time and pulse width are adjusted.(dynamic)

# property pulse\_transition

Control the pulse edge time (both rising and falling) in seconds (float).(dynamic)

# property pulse\_width

Control the pulse width in seconds (float).(dynamic)

# property ramp\_symmetry

Control the ramp waveform symmetry in percent (float, from 0 to 100).(dynamic)

# read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

# remove\_child(child)

Remove the child from the instrument and the corresponding collection.

#### **Parameters**

**child** – Instance of the child to delete.

#### reset()

Reset the instrument.

Return a property for the class based on the supplied commands. This property may be set, but raises an exception when being read from the instrument.

#### **Parameters**

- **set\_command** A string command that writes the value
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **check\_set\_errors** Toggles checking errors after setting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.

# property shape

Control the output waveform shape (str).(dynamic)

### shutdown()

Brings the instrument to a safe and stable state

# property square\_dutycycle

Control the square wave duty cycle in percent (float).(dynamic)

# property status

Get the status byte and Master Summary Status bit.

#### trigger()

Send a trigger signal to the function generator.

# property trigger\_source

Control the trigger source (str).

**values**(*command*, *separator='*, ', *cast=<class 'float'>*, *preprocess\_reply=None*, *maxsplit=-1*, \*\*kwargs)

Write a command to the instrument and return a list of formatted values from the result.

#### Parameters

- **command** SCPI command to be sent to the instrument.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.

- cast A type to cast each element of the splitted string.
- **\*\*kwargs** Keyword arguments to be passed to the *ask()* method.

### Returns

A list of the desired type, or strings where the casting fails.

# property voltage\_high

Control the upper voltage level in Volts (float).(dynamic)

# property voltage\_low

Control the lower voltage level in Volts (float).(dynamic)

wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

wait\_for\_trigger(timeout=3600, should\_stop=<function Agilent33500.<lambda>>)

Wait until the triggering has finished or timeout is reached.

#### **Parameters**

- **timeout** The maximum time the waiting is allowed to take. If timeout is exceeded, a TimeoutError is raised. If timeout is set to zero, no timeout will be used.
- should\_stop Optional function (returning a bool) to allow the waiting to be stopped before its end.

write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- command command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

class pymeasure.instruments.keysight.keysight81160A.Keysight81160AChannel(\*args, \*\*kwargs)

Bases: Agilent33500Channel

Represent the Keysight 81160A channel and provide a high-level interface for interacting with the instrument channels.

apply\_dc(voltage)

Apply a DC voltage.

#### **Parameters**

**voltage** – Voltage to be applied (float).

# apply\_noise(amplitude, offset)

Apply noise to the output.

### **Parameters**

- **amplitude** The amplitude of the noise (float).
- **offset** The offset voltage (float).

# apply\_pulse(frequency, amplitude, offset)

Apply a pulse waveform.

#### **Parameters**

- **frequency** Frequency of the pulse (float).
- amplitude Amplitude of the pulse (float).
- offset Offset voltage (float).

# apply\_sin(frequency, amplitude, offset)

Apply a sine waveform.

# **Parameters**

- **frequency** Frequency of the sine wave (float).
- **amplitude** Amplitude of the sine wave (float).
- offset Offset voltage (float).

# apply\_square(frequency, amplitude, offset)

Apply a square waveform.

# **Parameters**

- **frequency** Frequency of the square wave (float).
- amplitude Amplitude of the square wave (float).
- offset Offset voltage (float).

# apply\_user\_waveform(frequency, amplitude, offset)

Apply a user-defined waveform.

# **Parameters**

- **frequency** Frequency of the user waveform (float).
- amplitude Amplitude of the user waveform (float).
- offset Offset voltage (float).

# property coupling\_enabled

Control whether the channel coupling is enabled (bool). 'True' to copy values from this channel to another one.

#### delete\_waveform(name)

Delete a waveform from the generator's nonvolatile memory.

#### **Parameters**

**name** – The name of the user-defined waveform to be deleted.

# property free\_memory\_slots

Get the number of free non-volatile memory slots to store user waveforms (int).

#### property limit\_high

Control the high-level voltage limit in Volts (float).(dynamic)

# property limit\_low

Control the low-level voltage limit in Volts (float).(dynamic)

# property limit\_state\_enabled

Control whether the state limit is enabled (bool).

#### save\_waveform(waveform, name)

Save a waveform to the generator's nonvolatile memory.

#### **Parameters**

- waveform The waveform data.
- name The name that will be used to identify the waveform in memory, up to 12 characters. The first character must be a letter (A-Z), but the remaining characters can be numbers (0-9) or the underscore character ('\_'). Blank spaces are not allowed.

### property trigger\_count

Control the number of cycles to be output when a burst is triggered (int). Enable burst state if number > 1.

Short form of burst\_ncycles and burst\_state = True(dynamic)

# property trigger\_mode

Control the triggering mode (string, strictly 'IMM', 'INT2', 'EXT' or 'MAN').(dynamic)

#### property waveform\_volatile

Get volatile waveform data.

Warning: This property may not reflect the current device volatile waveform. It only returns the waveform previously set via this driver. If the device has been power-cycled, modified externally, or if no waveform was set, the returned value may be invalid or None.

#### Returns

waveform data (array-like) or None if not set.

#### property waveforms

Get the available user waveforms in memory (list[str]).

# 7.34 Kuhne Electronic

This section contains specific documentation on the Kuhne Electronic instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.34.1 KU SG 2.45 250 A - 2.45 GHz ISM-Band Microwave Generator

Bases: Instrument

Represents KU SG 2.45 250 A the 2.45 GHz ISM-Band Microwave Generator and provides a high-level interface for interacting with the instrument.

7.34. Kuhne Electronic 441

#### **Parameters**

**power\_limit** – power set-point limit in Watts (integer from 0 to 250). See *power\_setpoint* and *tune()*.

Usage example:

```
from pymeasure.instruments.kuhneelectronic import Kusg245_250A
generator = Kusg245_250A("ASRL3::INSTR", power_limit=100) # limits the output
                                                          # power set-point to 100 W
generator.external_enabled = False
                                       # biasing and RF output controlled by_
⇒serial comm
generator.power = 20
                                       # Sets the output power to 20 Watts
generator.bias_enabled = True
                                       # Enables amplifier biasing
generator.rf_enabled = True
                                       # Enables the RF output
p_fwd = generator.power_forward
                                       # Reads forward power in Watts
p_rev = generator.power_reverse
                                        # Reads reflected power in Watts
```

### property bias\_enabled

Control whether transistor biasing is enabled (boolean).

Biasing must be enabled before switching RF on (see  $rf_{enabled}$ ).

#### clear\_VSWR\_error()

Clear the VSWR error.

See: reflection limit.

# property external\_enabled

Control whether amplifier enabling is done via external inputs on 8-pin connector or via serial interface (boolean).

# property freq\_steps\_fine\_enabled

Control whether fine frequency steps are enabled (boolean).

#### property frequency\_coarse

Control coarse frequency in MHz (integer from 2400 to 2500).

Fine frequency mode must be disabled (see *freq\_steps\_fine\_enabled*). Resolution: 1 MHz. Invalid values are truncated.

# property frequency\_fine

Control fine frequency in kHz (integer from 2400000 to 2500000).

Fine frequency mode must be enabled (see *freq\_steps\_fine\_enabled*). Resolution: 10 kHz. Invalid values are truncated. Values are rounded to tens.

# property off\_time

Control off time for the pulse mode in ms (integer from 10 to 1000).

Resolution: 5 ms. Invalid values are truncated. Values are rounded to multipliers of 5.

# property phase\_shift

Control phase shift in degrees (float from 0 to 358.6).

Resolution: 8-bits. Values out of range are truncated.

#### property power\_forward

Measure forward power in Watts.

#### property power\_reverse

Measure reverse power in Watts.

#### property power\_setpoint

Control output power set-point in Watts (integer from 0 to power\_limit parameter - see constructor).

Resolution: 1 W. Invalid values are truncated. (dynamic)

#### property pulse\_mode\_enabled

Control whether pulse mode is enabled (boolean).



# 1 Note

Biasing must be enabled before the pulse mode is enabled (see bias\_enabled)

# property pulse\_width

Control pulse width in ms (integer from 10 to 1000).

Resolution: 5 ms. Invalid values are truncated. Values are rounded to multipliers of 5.

# property reflection\_limit

Control limit of reflection in Watts (integer in 0 - no limit, 100, 150, 180, 200, 230).



# 1 Note

If the limit for the reflected power is reached, the forward power is reduced to the specified value and the power control mechanism is locked until the alarm has been cleared by the user via clear\_VSWR\_error().

# property rf\_enabled

Control whether RF output is enabled (boolean).



# 1 Note

Biasing must be enabled before RF is enabled (see bias\_enabled)

# store\_settings()

Save actual settings to EEPROM.

The following parameters are stored: frequency mode (see freq\_steps\_fine\_enabled), frequency (see frequency\_coarse or frequency\_fine), output power set-point (see power\_setpoint), ON/OFF control setting (see external\_enabled), reflection limit (see reflection\_limit), on time for pulse mode (see pulse\_width) and off time for pulse mode (see off\_time).

# property temperature

Measure temperature near final transistor in °C.

#### tune(power)

Find and set frequency with lowest reflection at a given power.

7.34. Kuhne Electronic 443

#### **Parameters**

**power** – A power set-point for tuning (in Watts). (integer from 0 to power\_limit parameter - see constructor).

# turn\_off()

Safe turn-off the generator.

- 1. Disable RF output.
- 2. Deactivate biasing.

#### turn\_on()

Safe turn-on the generator.

- 1. Activate biasing.
- 2. Enable RF output.

# property version

Get firmware version.

#### property voltage\_32v

Measure 32V supply voltage in Volts.

# property voltage\_5v

Measure internal 5V supply voltage in Volts.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# 7.35 Lake Shore Cryogenics

This section contains specific documentation on the Lake Shore Cryogenics instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*. lakeshore3xx is a general class for Lake Shore temperature controllers of the series 3xx (like 325, 331, 334, 336, 340, 350).

# 7.35.1 Lake Shore 211 Temperature Monitor

Bases: SCPIUnknownMixin, Instrument

Represents the Lake Shore 211 Temperature Monitor and provides a high-level interface for interacting with the instrument.

Untested properties and methods will be noted in their docstrings.

```
controller = LakeShore211("GPIB::1")
print(controller.temperature_celsius) # Print the sensor temperature in celsius
```

**class AnalogMode**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntEnum

**class AnalogRange**(value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntEnum

**class RelayMode**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntEnum

**class RelayNumber**(value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntEnum

# property analog\_configuration

Control the analog mode and analog range. Values need to be supplied as a tuple of (analog mode, analog range) Analog mode can be 0 or 1

setting	mode
0	voltage
1	current

Analog range can be 0 through 5

setting	range
0	0 - 20  K
1	0 - 100  K
2	0 - 200  K
3	0 - 325  K
4	0 - 475  K
5	0 – 1000 K

# property analog\_out

Measure the percentage of output of the analog output.

**configure\_alarm**(*on=True*, *high\_value=270.0*, *low\_value=0.0*, *deadband=0*, *latch=False*)

Configures the alarm parameters for the input.

#### **Parameters**

- on Boolean setting of alarm, default True
- high\_value High value the temperature is checked against to activate the alarm
- low\_value Low value the temperature is checked against to activate the alarm
- deadband Value that the temperature must change outside of an alarm condition
- latch Specifies if the alarm should latch or not

#### configure\_relay(relay, mode)

Configure the relay mode of a relay

Property is UNTESTED

#### **Parameters**

- relay (RelayNumber) Specify which relay to configure
- mode (RelayMode) Specify which mode to assign

# property display\_units

Control the input data to display. Valid entries:

setting	units
'kelvin'	Kelvin
'celsius'	Celsius
'sensor'	Sensor Units
'fahrenheit'	Fahrenheit

# get\_alarm\_status()

Query the current alarm status

#### Returns

Dictionary of current status [on, high\_value, low\_value, deadband, latch]

# get\_relay\_mode(relay)

Get the status of a relay

Property is UNTESTED

### **Parameters**

relay (RelayNumber) – Specify which relay to query

#### Returns

Current RelayMode of queried relay

# reset\_alarm()

Resets the alarm of the Lakeshore 211

# property temperature\_celsius

Measure the temperature of the sensor in celsius

# property temperature\_fahrenheit

Measure the temperature of the sensor in fahrenheit

# property temperature\_kelvin

Measure the temperature of the sensor in kelvin

# property temperature\_sensor

Measure the temperature of the sensor in sensor units

# 7.35.2 Lake Shore 224 Temperature Monitor

Bases: SCPIUnknownMixin. Instrument

Represents the Lakeshore 224 Temperature monitor and provides a high-level interface for interacting with the instrument. Note that the 224 provides 12 temperature input channels (A, B, C1-5, D1-5). This driver makes use of the *LakeShore Channel Classes* 

```
monitor = LakeShore224('GPIB::1')
print(monitor.input_A.kelvin)
                               # Print the temperature in kelvin on sensor.
monitor.input_A.wait_for_temperature() # Wait for the temperature on sensor A to_
⇒stabilize.
input_0
       Channel
           LakeShoreTemperatureChannel
input_A
       Channel
           LakeShoreTemperatureChannel
input_B
       Channel
           LakeShoreTemperatureChannel
input_C1
       Channel
           LakeShoreTemperatureChannel
input_C2
       Channel
           LakeShoreTemperatureChannel
input_C3
       Channel
           LakeShoreTemperatureChannel
input_C4
       Channel
           LakeShoreTemperatureChannel
input_C5
       Channel
           LakeShoreTemperatureChannel
input_D1
       Channel
           LakeShoreTemperatureChannel
input_D2
       Channel
           LakeShoreTemperatureChannel
input_D3
       Channel
           LakeShoreTemperatureChannel
```

### input\_D4

#### Channel

LakeShoreTemperatureChannel

input\_D5

#### Channel

LakeShoreTemperatureChannel

# 7.35.3 Lake Shore 331 Temperature Controller

```
class pymeasure.instruments.lakeshore.LakeShore331(adapter, name='Lakeshore Model 331

Temperature Controller', read_termination=\r\n',

**kwargs)
```

Bases: SCPIMixin, Instrument

Represents the Lake Shore 331 Temperature Controller and provides a high-level interface for interacting with the instrument. Note that the 331 provides two input channels (A and B) and two output channels (1 and 2). This driver makes use of the *LakeShore Channel Classes*.

```
controller = LakeShore331("GPIB::1")

print(controller.output_1.setpoint)  # Print the current setpoint for loop 1
controller.output_1.setpoint = 50  # Change the loop 1 setpoint to 50 K
controller.output_1.heater_range = 'low'  # Change the heater range to low.
controller.input_A.wait_for_temperature()  # Wait for the temperature to stabilize.
print(controller.input_A.kelvin  # Print the temperature at sensor A.
```

Deprecated since version 0.16.0: Use LakeShore3xx instead.

# input\_A

# Channel

LakeShoreTemperatureChannel

input\_B

# Channel

LakeShoreTemperatureChannel

output\_1

# Channel

LakeShoreHeaterChannel

output\_2

#### Channel

LakeShoreHeaterChannel

# 7.35.4 Lake Shore 3xx Temperature Controller

Bases: SCPIMixin, Instrument

Represents a Lake Shore Temperature Controller of the 3xx-Series and provides a high-level interface for interacting with the instrument. Multiple input-and output options are implemented. Not all channels exist in all device versions! Some devices can host input option cards to add more input channels.

The LS 331 and 335 has two input channels (A and B) and two output channels (1 and 2). The LS 336 has four input channels (A, B, C and D) and four output channels (1, 2, 3 and 4).

The LS 340 has two input channels (A and B) and one output channel (1). Following extension cards can be used: \* 3465 Single Capacitance Input Option Card adds one channel (C). \* 3462 Dual Standard Input Option Card adds two channels (C and D). \* 3464 Dual Thermocouple Input Option Card adds two channels (C and D). \* 3468 Eight Channel Input Option Card adds eight channels (C1-C4, D1-D4).

The LS 350 has four input channels (A, B, C and D) and four output channels (1, 2, 3 and 4). Following extension card can be used: \* 3062 4-Channel Scanner option adds 4 additional channels (D2, D3, D4, and D5).

This driver makes use of the LakeShore Channel Classes.

```
controller = LakeShore3xx("GPIB::1")

print(controller.output_1.setpoint)  # Print the current setpoint for loop 1
controller.output_1.setpoint = 50  # Change the loop 1 setpoint to 50 K
controller.output_1.heater_range = 'low'  # Change the heater range to low.
controller.input_A.wait_for_temperature()  # Wait for the temperature to stabilize.
print(controller.input_A.kelvin)  # Print the temperature at sensor A.
```

# input\_A

#### Channel

LakeShoreTemperatureChannel

#### input\_B

# Channel

LakeShoreTemperatureChannel

# input\_C

#### Channel

LakeShoreTemperatureChannel

# input\_C1

# Channel

Lake Shore Temperature Channel

# input\_C2

#### Channel

LakeShoreTemperatureChannel

#### input\_C3

```
Channel
           LakeShoreTemperatureChannel
input_C4
        Channel
           LakeShoreTemperatureChannel
input_D
        Channel
           LakeShoreTemperatureChannel
input_D1
        Channel
           LakeShoreTemperatureChannel
input_D2
        Channel
           LakeShoreTemperatureChannel
input_D3
        Channel
           LakeShoreTemperatureChannel
input_D4
        Channel
           LakeShoreTemperatureChannel
input_D5
        Channel
           LakeShoreTemperatureChannel
output_1
        Channel
           LakeShoreHeaterChannel
output_2
        Channel
           LakeShoreHeaterChannel
output_3
```

Channel

Channel

output\_4

LakeShoreHeaterChannel

LakeShoreHeaterChannel

# 7.35.5 Lake Shore 421 Gaussmeter

Bases: Instrument

Represents the Lake Shore 421 Gaussmeter and provides a high-level interface for interacting with the instrument.

```
gaussmeter = LakeShore421("COM1")
gaussmeter.unit = "T"  # Set units to Tesla
gaussmeter.auto_range = True  # Turn on auto-range
gaussmeter.fast_mode = True  # Turn on fast-mode
```

A delay of 50 ms is ensured between subsequent writes, as the instrument cannot correctly handle writes any faster

### property alarm\_active

Get whether the alarm is triggered.

# property alarm\_audible

Control the audible alarm beeper.

# property alarm\_high

Control the upper setpoint for the alarm mode in the current units. This takes into account the field multiplier.

# property alarm\_high\_multiplier

Get the multiplier for the upper alarm setpoint field.

# property alarm\_high\_raw

Control the upper setpoint for the alarm mode in the current unit and multiplier.

# property alarm\_in\_out

Control whether an active alarm is caused when the field reading is inside ("Inside") or outside ("Outside") of the high and low setpoint values.

# property alarm\_low

Control the lower setpoint for the alarm mode in the current units. This takes into account the field multiplier.

# property alarm\_low\_multiplier

Get the multiplier for the lower alarm setpoint field.

# property alarm\_low\_raw

Control the lower setpoint for the alarm mode in the current units and multiplier.

### property alarm\_mode\_enabled

Control the alarm mode.

# property alarm\_sort\_enabled

Control the alarm Sort Pass/Fail function.

#### property auto\_range

Control the auto-range option of the meter. Valid values are True and False. Note that the auto-range is relatively slow and might not suffice for rapid measurements.

# property display\_filter\_enabled

Control the display filter to make it more readable when the probe is exposed to a noisy field. The filter function makes a linear average of 8 readings and settles in approximately 2 seconds.

#### property fast\_mode

Control the fast-mode option of the meter. Valid values are True and False. When enabled, the relative mode, Max Hold mode, alarms, and autorange are disabled.

# property field

Get the field in the current units. This property takes into account the field multiplier. Get np.nan if field is out of range.

### property field\_mode

Control whether the gaussmeter measures AC or DC magnetic fields. Valid values are "AC" and "DC".

# property field\_multiplier

Get the field multiplier for the returned magnetic field.

# property field\_range

Control (floating) the field range of the meter in the current unit (G or T). Valid values are 30e3, 3e3, 300, 30 (when in Gauss), or 0.003, 0.03, 0.3, and 3 (when in Tesla).

# property field\_range\_raw

Control (integer) the field range of the meter. Valid values are 0 (highest) to 3 (lowest).

# property field\_raw

Get the field in the current units and multiplier

# property front\_panel\_brightness

Control (integer) the brightness of the from panel display. Valid values are 0 (dimmest) to 7 (brightest).

# property front\_panel\_locked

Control the lock state of all front panel entries except pressing the Alarm key to silence alarms.

#### property max\_hold\_enabled

Control the Max Hold function to store the largest field since the last reset (with max\_hold\_reset).

# property max\_hold\_field

Get the largest field since the last reset in the current units. This property takes into account the field multiplier. Returns np.nan if field is out of range.

#### property max\_hold\_field\_raw

Get the largest field since the last reset in the current units and multiplier.

# property max\_hold\_multiplier

Get the multiplier for the returned max hold field.

# max\_hold\_reset()

Clears the stored Max Hold value.

#### property probe\_type

Get type of field-probe used with the gaussmeter. Possible values are High Sensitivity, High Stability, or Ultra-High Sensitivity.

# property relative\_field

Get the relative field in the current units. This property takes into account the field multiplier. Returns np.nan if field is out of range.

#### property relative\_field\_raw

Get the relative field in the current units and the current multiplier.

### property relative\_mode\_enabled

Control the relative mode to see small variations with respect to a given setpoint.

#### property relative\_multiplier

Get the relative field multiplier for the returned magnetic field.

# property relative\_setpoint

Control the setpoint for the relative field mode in the current units. This takes into account the field multiplier.

# property relative\_setpoint\_multiplier

Get the multiplier for the setpoint field.

# property relative\_setpoint\_raw

Control the setpoint for the relative field mode in the current units and multiplier.

# property serial\_number

Get the serial number of the probe.

#### shutdown()

Close the serial connection to the system.

#### property unit

Control (string) the units used by the gaussmeter. Valid values are G (Gauss), T (Tesla).

#### write(command)

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- command command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# zero\_probe(wait=True)

Reset the probe value to 0. It is normally used with a zero gauss chamber, but may also be used with an open probe to cancel the Earth magnetic field. To cancel larger magnetic fields, the relative mode should be used.

#### **Parameters**

wait (boo1) – Wait for 20 seconds after issuing the command to allow the resetting to finish.

# 7.35.6 Lake Shore 425 Gaussmeter

Bases: Instrument

Represents the LakeShore 425 Gaussmeter and provides a high-level interface for interacting with the instrument

To allow user access to the LakeShore 425 Gaussmeter in Linux, create the file: /etc/udev/rules.d/52-lakeshore425.rules, with contents:

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="1fb9", ATTRS{idProduct}=="0401", MODE="0666", SYMLINK+="lakeshore425"
```

Then reload the udev rules with:

```
sudo udevadm control --reload-rules sudo udevadm trigger
```

The device will be accessible through /dev/lakeshore425.

# ac\_mode(wideband=True)

Sets up a measurement of an oscillating (AC) field

#### auto\_range()

Sets the field range to automatically adjust

# dc\_mode(wideband=True)

Sets up a steady-state (DC) measurement of the field

# property field

Get the field in the current units

measure(points, has\_aborted=<function LakeShore425.<lambda>>, delay=0.001)

Returns the mean and standard deviation of a given number of points while blocking

#### property mode

Control the mode, filter, and bandwidth settings.

#### property range

Control the field range in units of Gauss, which can take the values 35, 350, 3500, and 35,000 G. (float)

# property unit

Control the units of the instrument, which can take the values of G, T, Oe, or A/m. (str)

# zero\_probe()

Initiates the zero field sequence to calibrate the probe

#### 7.35.7 LakeShore Channel Classes

Several Lakeshore instruments are channel based and make use of the *Channel Interface*. For temperature monitoring and controller instruments the following common *Channel Classes* are utilized:

Bases: Channel

Temperature input channel on a lakeshore temperature monitor. Reads the temperature in kelvin, celsius, or sensor units. Also provides a method to block the program until a given stable temperature is reached.

### property celcius

Get celsius attribute with celcius (sic) property.

Deprecated since version 0.14.0: Use celsius instead.

#### property celsius

Get the temperature in celsius from a channel.

# property kelvin

Get the temperature in kelvin from a channel.

#### property sensor

Get the temperature in sensor units from a channel.

Blocks the program, waiting for the temperature to reach the target within the accuracy (%), checking this each interval time in seconds.

#### **Parameters**

- target Target temperature in kelvin, celsius, or sensor units.
- unit 'kelvin', 'celsius', or 'sensor' specifying the unit for queried temperature values.
- accuracy An acceptable percentage deviation between the target and temperature.
- **interval** Interval time in seconds between queries.
- timeout A timeout in seconds after which an exception is raised
- **should\_stop** A function that returns True if waiting should stop, by default this always returns False

class pymeasure.instruments.lakeshore.lakeshore\_base.LakeShoreHeaterChannel(parent, id)

Bases: Channel

Heater output channel on a lakeshore temperature controller. Provides properties to query the output power in percent of the max, set the manual output power, heater range, and PID temperature setpoint.

#### property mout

Control manual heater output in percent.

#### property output

Get the heater output in percent of the max.

#### property range

Control heater range, which can take the values: off, low, medium, and high.

# property setpoint

Control the setpoint temperature in the preferred units of the control loop sensor.

# 7.36 LeCroy

This section contains specific documentation on the LeCroy instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

If the instrument you are looking for is not here, also check *Teledyne* for newer instruments.

7.36. LeCroy 455

# 7.36.1 LeCroy T3DSO1204 Oscilloscope

Bases: TeledyneOscilloscope

Represents the LeCroy T3DSO1204 Oscilloscope interface for interacting with the instrument.

Refer to the LeCroy T3DSO1204 Oscilloscope Programmer's Guide for further details about using the lower-level methods to interact directly with the scope.

This implementation is based on the shared base class TeledyneOscilloscope.

Attributes:

WRITE\_INTERVAL\_S after the previous one, the code blocks until at least WRITE\_INTERVAL\_S seconds have passed. Because the oscilloscope takes a non-negligible time to perform some operations, it might be needed for the user to tweak the sleep time between commands. The WRITE\_INTERVAL\_S is set to 10ms as default however its optimal value heavily depends on the actual commands and on the connection type, so it is impossible to give a unique value to fit all cases. An interval between 10ms and 500ms second proved to be good, depending on the commands and connection latency.

```
scope = LeCroyT3DS01204(resource)
scope.autoscale()
ch1_data_array, ch1_preamble = scope.download_waveform(source="C1", points=2000)
# ...
scope.shutdown()
```

ch\_1

# Channel

LeCroyT3DS01204Channel

 $ch_2$ 

#### Channe

LeCroyT3DS01204Channel

 $ch_3$ 

#### Channel

LeCroyT3DS01204Channel

ch\_4

#### Channel

LeCroyT3DS01204Channel

class BaseChannelCreator(cls, \*\*kwargs)

Bases: object

Base class for ChannelCreator and MultiChannelCreator.

#### **Parameters**

- **cls** Class for all children or tuple/list of classes, one for each child.
- **\*\*kwargs** Keyword arguments for all children.

#### class ChannelCreator(cls, id=None, \*\*kwargs)

Bases: BaseChannelCreator

Add a single channel to the parent class.

The child will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name that ChannelCreator was assigned to in the *Instrument* class will be the name of the channel interface.

```
class Extreme5000(Instrument):
    # Two output channels, accessible by their property names
    # and both are accessible through the 'channels' collection
    output_A = Instrument.ChannelCreator(Extreme5000Channel, "A")
    output_B = Instrument.ChannelCreator(Extreme5000Channel, "B")
    # A channel without a channel accessible through the 'motor' collection
    motor = Instrument.ChannelCreator(MotorControl)

inst = SomeInstrument()
# Set the extreme_temp for channel A of Extreme5000 instrument
inst.output_A.extreme_temp = 42
```

#### **Parameters**

- cls Channel class for channel interface
- id The id of the channel on the instrument, integer or string.
- \*\*kwargs Keyword arguments for all children.

# class MultiChannelCreator(cls, id=None, prefix='ch\_', \*\*kwargs)

Bases: BaseChannelCreator

Add channels to the parent class.

The children will be added to the parent instance at instantiation with CommonBase.add\_child(). The attribute name (e.g. channels) will be used as the *collection* of the children. You may define the attribute prefix. If there are no other pressing reasons, use channels as the attribute name and leave the prefix at the default "ch\_".

# **Parameters**

- cls Class for all children or tuple/list of classes, one for each child.
- id tuple/list of ids of the channels on the instrument.
- **prefix** Collection prefix for the attributes, e.g. "*ch*\_" creates attribute *self.ch*\_A. If prefix evaluates False, the child will be added directly under the variable name. Required if id is tuple/list.

7.36. LeCroy 457

• **\*\*kwargs** – Keyword arguments for all children.

# property acquisition\_average

Control the averaging times of average acquisition.

# acquisition\_sample\_size(source)

Get acquisition sample size for a certain channel. Used mainly for waveform acquisition. If the source is MATH, the SANU? MATH query does not seem to work, so I return the memory size instead.

#### **Parameters**

**source** – channel number of channel name.

#### Returns

acquisition sample size of that channel.

# property acquisition\_sample\_size\_c1

Get the number of data points that the hardware will acquire from the input signal of channel 1. Note. Channel 2 and channel 1 share the same ADC, so the sample is the same too.

# property acquisition\_sample\_size\_c2

Get the number of data points that the hardware will acquire from the input signal of channel 2. Note. Channel 2 and channel 1 share the same ADC, so the sample is the same too.

# property acquisition\_sample\_size\_c3

Get the number of data points that the hardware will acquire from the input signal of channel 3. Note. Channel 3 and channel 4 share the same ADC, so the sample is the same too.

# property acquisition\_sample\_size\_c4

Get the number of data points that the hardware will acquire from the input signal of channel 4. Note. Channel 3 and channel 4 share the same ADC, so the sample is the same too.

### property acquisition\_sampling\_rate

Get the sample rate of the scope.

#### property acquisition\_status

Get the acquisition status of the scope.

#### property acquisition\_type

Control the type of data acquisition.

Can be 'normal', 'peak', 'average', 'highres'.

#### add\_child(cls, id=None, collection='channels', prefix='ch', attr name='', \*\*kwargs)

Add a child to this instance and return its index in the children list.

The newly created child may be accessed either by the id in the children dictionary or by the created attribute, e.g. the fifth channel of *instrument* with id "F" has two access options: instrument.channels["F"] == instrument.ch\_F



# Note

Do not change the default *collection* or *prefix* parameter, unless you have to distinguish several collections of different children, e.g. different channel types (analog and digital).

#### **Parameters**

• **cls** – Class of the channel.

- **id** Child id how it is used in communication, e.g. "A".
- collection Name of the collection of children, used for dictionary access to the channel interfaces.
- **prefix** For creating multiple channel interfaces, the prefix e.g. "ch\_" is prepended to the attribute name of the channel interface self.ch\_A. If prefix evaluates False, the child will be added directly under the collection name.
- attr\_name For creating a single channel interface, the attr\_name argument is used when setting the attribute name of the channel interface.
- **\*\*kwargs** Keyword arguments for the channel creator.

#### Returns

Instance of the created child.

# ask(command, query\_delay=None)

Write a command to the instrument and return the read response.

#### **Parameters**

- **command** Command string to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.

#### Returns

String returned by the device without read\_termination.

#### autoscale()

Autoscale displayed channels.

# binary\_values(command, query\_delay=None, \*\*kwargs)

Write a command to the instrument and return a numpy array of the binary data.

#### **Parameters**

- **command** Command to be sent to the instrument.
- query\_delay Delay between writing and reading in seconds.
- **kwargs** Arguments for read\_binary\_values().

# Returns

NumPy array of values.

#### property bwlimit

Set the internal low-pass filter for all channels.(dynamic)

# center\_trigger()

Set the trigger levels to center of the trigger source waveform.

#### ch(source)

Get channel object from its index or its name. Or if source is "math", just return the scope object.

#### **Parameters**

```
source – can be 1, 2, 3, 4 or C1, C2, C3, C4, MATH
```

# Returns

handle to the selected source.

7.36. LeCroy 459

#### check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

# check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# clear()

Clear the instrument status byte.

#### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

Return a property for the class based on the supplied commands. This property may be set and read from the instrument. See also *measurement()* and *setting()*.

#### **Parameters**

- **get\_command** A string command that asks for the value, set to *None* if get is not supported (see also *setting()*).
- **set\_command** A string command that writes the value, set to *None* if set is not supported (see also *measurement* ()).
- docs A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map

- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that takes a command and allows processing before executing the command

Deprecated since version 0.12: Use a dynamic property instead.

- **check\_set\_errors** Toggles checking errors after setting
- **check\_get\_errors** Toggles checking errors after getting
- dynamic Specify whether the property parameters are meant to be changed in instances or subclasses.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- **maxsplit** The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- values\_kwargs (dict) Further keyword arguments for values().
- \*\*kwargs Keyword arguments for values().

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

Example of usage of dynamic parameter is as follows:

```
class GenericInstrument(Instrument):
    center_frequency = Instrument.control(
        ":SENS:FREQ:CENT?;", ":SENS:FREQ:CENT %e GHz;",
        " A floating point property that represents the frequency ... ",
        validator=strict_range,
        # Redefine this in subclasses to reflect actual instrument value:
        values=(1, 20),
        dynamic=True # enable changing property parameters on-the-fly
)

class SpecificInstrument(GenericInstrument):
    # Identical to GenericInstrument, except for frequency range
    # Override the "values" parameter of the "center_frequency" property
    center_frequency_values = (1, 10) # Redefined at subclass level

instrument = SpecificInstrument()
instrument.center_frequency_values = (1, 6e9) # Redefined at instance level
```

# Warning

Unexpected side effects when using dynamic properties

7.36. LeCroy 461

Users must pay attention when using dynamic properties, since definition of class and/or instance attributes matching specific patterns could have unwanted side effect. The attribute name pattern *property\_param*, where *property* is the name of the dynamic property (e.g. *center\_frequency* in the example) and *param* is any of this method parameters name except *dynamic* and *docs* (e.g. *values* in the example) has to be considered reserved for dynamic property control.

#### default\_setup()

Set up the oscilloscope for remote operation.

The COMM\_HEADER command controls the way the oscilloscope formats response to queries. This command does not affect the interpretation of messages sent to the oscilloscope. Headers can be sent in their long or short form regardless of the CHDR setting. By setting the COMM\_HEADER to OFF, the instrument is going to reply with minimal information, and this makes the response message much easier to parse. The user should not be fiddling with the COMM\_HEADER during operation, because if the communication header is anything other than OFF, the whole driver breaks down.

# display\_parameter(parameter, channel)

Same as the display\_parameter method in the Channel subclass.

### download image()

Get a BMP image of oscilloscope screen in bytearray of specified file format.

#### download\_waveform(source, requested points=None, sparsing=None)

Get data points from the specified source of the oscilloscope.

The returned objects are two np.ndarray of data and time points and a dict with the waveform preamble, that contains metadata about the waveform.

#### **Parameters**

- source measurement source. It can be "C1", "C2", "C3", "C4", "MATH".
- **requested\_points** number of points to acquire. If None the number of points requested in the previous call will be assumed, i.e. the value of the number of points stored in the oscilloscope memory. If 0 the maximum number of points will be returned.
- **sparsing** interval between data points. For example if sparsing = 4, only one point every 4 points is read. If 0 or None the sparsing of the previous call is assumed, i.e. the value of the sparsing stored in the oscilloscope memory.

#### Returns

data\_ndarray, time\_ndarray, waveform\_preamble\_dict: see waveform\_preamble property for dict format.

# static get\_channel\_pairs(cls)

Return a list of all the Instrument's channel pairs

# static get\_channels(cls)

Return a list of all the Instrument's ChannelCreator and MultiChannelCreator instances

# property grid\_display

Control the type of the grid which is used to display (FULL, HALF, OFF).

#### property id

Get the identification of the instrument.

# property intensity

Set the intensity level of the grid or the trace in percent

#### property math\_define

Control the desired waveform math operation between two channels.

Three parameters must be passed as a tuple:

- 1. source 1: source channel on the left
- 2. operation : operator must be "\*", "/", "+", "-"
- 3. source2: source channel on the right

### property math\_vdiv

Control the vertical scale of the selected math operation.

This command is only valid in add, subtract, multiply and divide operation. Note: legal values for the scale depend on the selected operation.

### property math\_vpos

Control the vertical position of the math waveform with specified source.

Note: the point represents the screen pixels and is related to the screen center. For example, if the point is 50. The math waveform will be displayed 1 grid above the vertical center of the screen. Namely one grid is 50.

#### property measure\_delay

Control measurement delay.

The MEASURE\_DELY command places the instrument in the continuous measurement mode and starts a type of delay measurement. The MEASURE\_DELY? query returns the measured value of delay type. The command accepts three arguments with the following syntax:

```
measure_delay = (<type>,<sourceA>,<sourceB>)
<type> := {PHA,FRR,FRF,FFR,FFF,LRR,LRF,LFR,LFF,SKEW}
<sourceA>,<sourceB> := {C1,C2,C3,C4} where if sourceA=CX and sourceB=CY, then X < Y</pre>
```

Туре	Description
PHA	The phase difference between two channels. (rising edge - rising edge)
FRR	Delay between two channels. (first rising edge - first rising edge)
FRF	Delay between two channels. (first rising edge - first falling edge)
FFR	Delay between two channels. (first falling edge - first rising edge)
FFF	Delay between two channels. (first falling edge - first falling edge)
LRR	Delay between two channels. (first rising edge - last rising edge)
LRF	Delay between two channels. (first rising edge - last falling edge)
LFR	Delay between two channels. (first falling edge - last rising edge)
LFF	Delay between two channels. (first falling edge - last falling edge)
Skew	Delay between two channels. (edge – edge of the same type)

### measure\_parameter(parameter, channel)

Same as the measure\_parameter method in the Channel subclass

```
static measurement(get_command, docs, values=(), map_values=False, get_process=<function

CommonBase.<lambda>>, get_process_list=<function CommonBase.<lambda>>,

command_process=None, check_get_errors=False, dynamic=False,

preprocess_reply=None, separator=', ', maxsplit=-1, cast=<class 'float'>,

values_kwargs=None, **kwargs)
```

7.36. LeCroy 463

Return a property for the class based on the supplied commands. This is a measurement quantity that may only be read from the instrument, not set.

#### **Parameters**

- **get\_command** A string command that asks for the value
- **docs** A docstring that will be included in the documentation
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **get\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **get\_process\_list** A function that takes the value list and processes it.
- **command\_process** A function that take a command and allows processing before executing the command, for getting

Deprecated since version 0.12: Use a dynamic property instead.

- check\_get\_errors Toggles checking errors after getting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- **maxsplit** The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- **values\_kwargs** (*dict*) Further keyword arguments for *values*().
- \*\*kwargs Keyword arguments for *values*().

Deprecated since version 0.12: Use *values\_kwargs* dictionary parameter instead.

### property memory\_size

Control the maximum depth of memory.

<size>:={7K,70K,700K,7M} for non-interleaved mode. Non-interleaved means a single channel is active per A/D converter. Most oscilloscopes feature two channels per A/D converter.

<size>:={14K,140K,1.4M,14M} for interleave mode. Interleave mode means multiple active channels per A/D converter.

# property menu

Control the bottom menu enabled state (strict bool).

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use *check\_errors()* instead.

### property options

Get the device options installed.

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

Read up to (excluding) read\_termination or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

```
read_bytes(count, **kwargs)
```

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

#### **Returns bytes**

Bytes response of the instrument (including termination).

### remove\_child(child)

Remove the child from the instrument and the corresponding collection.

#### **Parameters**

**child** – Instance of the child to delete.

#### reset()

Reset the instrument.

#### run()

Starts repetitive acquisitions.

This is the same as pressing the Run key on the front panel.

Return a property for the class based on the supplied commands. This property may be set, but raises an exception when being read from the instrument.

#### **Parameters**

- **set\_command** A string command that writes the value
- **docs** A docstring that will be included in the documentation
- **validator** A function that takes both a value and a group of valid values and returns a valid value, while it otherwise raises an exception
- values A list, tuple, range, or dictionary of valid values, that can be used as to map values if map\_values is True.
- map\_values A boolean flag that determines if the values should be interpreted as a map
- **set\_process** A function that takes a value and allows processing before value mapping, returning the processed value
- **check\_set\_errors** Toggles checking errors after setting
- **dynamic** Specify whether the property parameters are meant to be changed in instances or subclasses. See *control()* for an usage example.

7.36. LeCroy 465

#### shutdown()

Brings the instrument to a safe and stable state

#### single()

Causes the instrument to acquire a single trigger of data.

This is the same as pressing the Single key on the front panel.

#### property status

Get the status byte and Master Summary Status bit.

#### stop()

Stops the acquisition. This is the same as pressing the Stop key on the front panel.

### property timebase

Get timebase setup as a dict containing the following keys:

- "timebase\_scale": horizontal scale in seconds/div (float)
- "timebase\_offset": interval in seconds between the trigger and the reference position (float)
- "timebase\_hor\_magnify": horizontal scale in the zoomed window in seconds/div (float)
- "timebase\_hor\_position": horizontal position in the zoomed window in seconds (float)

#### property timebase\_hor\_magnify

Control the zoomed (delayed) window horizontal scale (seconds/div).

The main sweep scale determines the range for this command.

### property timebase\_hor\_position

Control the horizontal position in the zoomed (delayed) view of the main sweep.

The main sweep range and the main sweep horizontal position determine the range for this command. The value for this command must keep the zoomed view window within the main sweep range.

### property timebase\_offset

Control the time interval in seconds between the trigger event and the reference position (at center of screen by default).

### property timebase\_scale

Control the horizontal scale (units per division) in seconds for the main window (float).

### timebase\_setup(scale=None, offset=None, hor\_magnify=None, hor\_position=None)

Set up timebase. Unspecified parameters are not modified. Modifying a single parameter might impact other parameters. Refer to oscilloscope documentation and make multiple consecutive calls to timebase setup if needed.

#### **Parameters**

- **scale** interval in seconds between the trigger event and the reference position.
- offset horizontal scale per division in seconds/div.
- **hor\_magnify** horizontal scale in the zoomed window in seconds/div.
- **hor\_position** horizontal position in the zoomed window in seconds.

### property trigger

Get trigger setup as a dict containing the following keys:

• "mode": trigger sweep mode [auto, normal, single, stop]

- "trigger\_type": condition that will trigger the acquisition of waveforms [edge, slew,glit,intv,runt,drop]
- "source": trigger source [c1,c2,c3,c4]
- "hold\_type": hold type (refer to page 172 of programing guide)
- "hold\_value1": hold value1 (refer to page 172 of programing guide)
- "hold\_value2": hold value2 (refer to page 172 of programing guide)
- "coupling": input coupling for the selected trigger sources
- "level": trigger level voltage for the active trigger source
- "level2": trigger lower level voltage for the active trigger source (only slew/runt trigger)
- "slope": trigger slope of the specified trigger source

#### property trigger\_mode

Control the trigger sweep mode (string).

<mode>:= {AUTO,NORM,SINGLE,STOP}

- auto: When AUTO sweep mode is selected, the oscilloscope begins to search for the trigger signal that meets the conditions. If the trigger signal is satisfied, the running state on the top left corner of the user interface shows Trig'd, and the interface shows stable waveform. Otherwise, the running state always shows Auto, and the interface shows unstable waveform.
- normal: When NORMAL sweep mode is selected, the oscilloscope enters the wait trigger state and begins to search for trigger signals that meet the conditions. If the trigger signal is satisfied, the running state shows Trig'd, and the interface shows stable waveform. Otherwise, the running state shows Ready, and the interface displays the last triggered waveform (previous trigger) or does not display the waveform (no previous trigger).
- single: When SINGLE sweep mode is selected, the backlight of SINGLE key lights up, the oscilloscope enters the waiting trigger state and begins to search for the trigger signal that meets the conditions. If the trigger signal is satisfied, the running state shows Trig'd, and the interface shows stable waveform. Then, the oscilloscope stops scanning, the RUN/STOP key is red light, and the running status shows Stop. Otherwise, the running state shows Ready, and the interface does not display the waveform.
- stopped: STOP is a part of the option of this command, but not a trigger mode of the oscilloscope.

# property trigger\_select

Control the condition that will trigger the acquisition of waveforms (string).

Depending on the trigger type, additional parameters must be specified. These additional parameters are grouped in pairs. The first in the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs may be given in any order and restricted to those variables to be changed.

There are five parameters that can be specified. Parameters 1. 2. 3. are always mandatory. Parameters 4. 5. are required only for certain combinations of the previous parameters.

- 1. <trig\_type>:={edge, slew, glit, intv, runt, drop}
- 2. <source>:={c1, c2, c3, c4, line}
- 3. <hold type>:=
  - {ti, off} for edge trigger.
  - {ti} for drop trigger.
  - {ps, pl, p2, p1} for glit/runt trigger.

7.36. LeCroy 467

- {is, il, i2, i1} for slew/intv trigger.
- 4. <hold\_value1>:= a time value with unit.
- 5. <hold\_value2>:= a time value with unit.

#### Note:

- "line" can only be selected when the trigger type is "edge".
- All time arguments should be given in multiples of seconds. Use the scientific notation if necessary.
- The range of hold\_values varies from trigger types. [80nS, 1.5S] for "edge" trigger, and [2nS, 4.2S] for others.
- The trigger\_select command is switched automatically between the short, normal and extended version depending on the number of expected parameters.

Set up trigger.

Unspecified parameters are not modified. Modifying a single parameter might impact other parameters. Refer to oscilloscope documentation and make multiple consecutive calls to trigger\_setup and channel\_setup if needed.

#### **Parameters**

- **mode** trigger sweep mode [auto, normal, single, stop]
- **source** trigger source [c1, c2, c3, c4, line]
- **trigger\_type** condition that will trigger the acquisition of waveforms [edge,slew,glit,intv,runt,drop]
- **hold\_type** hold type (refer to page 172 of programing guide)
- **hold\_value1** hold value1 (refer to page 172 of programing guide)
- **hold\_value2** hold value2 (refer to page 172 of programing guide)
- coupling input coupling for the selected trigger sources
- level trigger level voltage for the active trigger source
- level2 trigger lower level voltage for the active trigger source (only slew/runt trigger)
- **slope** trigger slope of the specified trigger source

**values**(*command*, *separator='*, ', *cast=<class 'float'>*, *preprocess\_reply=None*, *maxsplit=-1*, \*\*kwargs)
Write a command to the instrument and return a list of formatted values from the result.

#### **Parameters**

- **command** SCPI command to be sent to the instrument.
- **preprocess\_reply** Optional callable used to preprocess the string received from the instrument, before splitting it. The callable returns the processed string.
- **separator** A separator character to split the string returned by the device into a list.
- maxsplit The string returned by the device is splitted at most *maxsplit* times. -1 (default) indicates no limit.
- cast A type to cast each element of the splitted string.
- \*\*kwargs Keyword arguments to be passed to the ask() method.

#### Returns

A list of the desired type, or strings where the casting fails.

### wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

#### property waveform\_first\_point

Control the address of the first data point to be sent (int). For waveforms acquired in sequence mode, this refers to the relative address in the given segment. The first data point starts at zero and is strictly positive.

### property waveform\_points

Control the number of waveform points to be transferred with the digitize method (int). NP = 0 sends all data points.

Note that the oscilloscope may provide less than the specified nb of points.

### property waveform\_preamble

Get preamble information for the selected waveform source as a dict with the following keys:

- "type": normal, peak detect, average, high resolution (str)
- "requested\_points": number of data points requested by the user (int)
- "sampled\_points": number of data points sampled by the oscilloscope (int)
- "transmitted points": number of data points actually transmitted (optional) (int)
- "memory\_size": size of the oscilloscope internal memory in bytes (int)
- "sparsing": sparse point. It defines the interval between data points. (int)
- "first\_point": address of the first data point to be sent (int)
- "source": source of the data: "C1", "C2", "C3", "C4", "MATH".
- "unit": Physical units of the Y-axis
- "type": type of data acquisition. Can be "normal", "peak", "average", "highres"
- "average": average times of average acquisition
- "sampling\_rate": sampling rate (it is a read-only property)
- "grid\_number": number of horizontal grids (it is a read-only property)
- "status": acquisition status of the scope. Can be "stopped", "triggered", "ready", "auto", "armed"
- "xdiv": horizontal scale (units per division) in seconds
- "xoffset": time interval in seconds between the trigger event and the reference position
- "ydiv": vertical scale (units per division) in Volts
- "yoffset": value that is represented at center of screen in Volts

### property waveform\_sparsing

Control the interval between data points (integer). For example:

SP = 0 sends all data points. SP = 4 sends 1 point every 4 data points.

7.36. LeCroy 469

### write(command, \*\*kwargs)

Write the command to the instrument through the adapter.

Note: if the last command was sent less than WRITE\_INTERVAL\_S before, this method blocks for the remaining time so that commands are never sent with rate more than 1/WRITE\_INTERVAL\_S Hz.

#### **Parameters**

**command** – command string to be sent to the instrument

#### write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

### class pymeasure.instruments.lecroy.lecroyT3DS01204.LeCroyT3DS01204Channel(parent, id)

Bases: TeledyneOscilloscopeChannel

Implementation of a LeCroy T3DSO1204 Oscilloscope channel.

Implementation modeled on Channel object of Keysight DSOX1102G instrument.

### property bwlimit

Control the 20 MHz internal low-pass filter (strict bool).

This oscilloscope only has one frequency available for this filter.

### property invert

Control the inversion of the input signal (strict bool).

#### property skew\_factor

Control the channel-to-channel skew factor for the specified channel. Each analog channel can be adjusted + or -100 ns for a total of 200 ns difference between channels. You can use the oscilloscope's skew control to remove cable-delay errors between channels.

### property trigger\_level2

Control the lower trigger level voltage for the specified source (float). Higher and lower trigger levels are used with runt/slope triggers. When setting the trigger level it must be divided by the probe attenuation. This is not documented in the datasheet and it is probably a bug of the scope firmware. An out-of-range value will be adjusted to the closest legal value.

### property unit

Control the unit of the specified trace. Measurement results, channel sensitivity, and trigger level will reflect the measurement units you select. ("A" for Amperes, "V" for Volts).

# 7.37 MKS Instruments

This section contains specific documentation on the MKS Instruments devices that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.37.1 MKS Instruments Abstract Instrument

Bases: Instrument

Abstract MKS Instrument

Connection to the device is made through an RS232/RS485 serial connection. The communication protocol of these devices is as follows:

Query: '@<aaa><Command>?;FF' with the response '@<aaa>ACK<Response>;FF' Set command: '@<aaa><Command>!<parameter>;FF' with the response '@<aaa>ACK<Response>;FF' Above <aaa> is an address from 001 to 254 which can be specified upon initialization. Since ';FF' is not supported by pyvisa as terminator this class overloads the device communication methods.

#### **Parameters**

- adapter pyvisa resource name of the instrument or adapter instance
- **name** (*string*) The name of the instrument.
- address device address included in every message to the instrument (default=253)
- kwargs Any valid key-word argument for Instrument

### check\_set\_errors()

Check reply string for acknowledgement string.

#### read()

Reads from the instrument including the correct termination characters

#### write(command)

Write to the instrument including the device address.

#### **Parameters**

**command** – command string to be sent to the instrument

class pymeasure.instruments.mksinst.mksinst.RelayChannel(parent, id)

Bases: Channel

Settings of the optionally included setpoint relay.

The relay is energized either below or above the setpoint depending on the 'direction' property. The relay is de-energized when the reset value is crossed in the opposite direction.

Note that device by default uses an auto hysteresis setting of 10% of the setpoint value that overwrites the current reset value whenever the setpoint value or direction is changed. If other hysteresis value than 10% is required, first set the setpoint value and direction before setting the reset value.

### property direction

Control the switching direction

7.37. MKS Instruments 471

#### property resetpoint

Control the relay switch off value

### property setpoint

Control the relay switch setpoint

#### property status

Get the setpoint relay status

# 7.37.2 MKS Instruments 937B Vacuum Gauge Controller

Bases: MKSInstrument

MKS 937B vacuum gauge controller

Connection to the device is made through an RS232/RS485 serial connection.

The 937B gauge controller can connect up to 6 pressure measurement channels for gauges of various types which includes ionization gauges, Pirani and piezo gauges. Based on the pressure values of the gauges twelve setpoint relays can be energized when certain pressure values are reached. The assignment of the relays to measurement channels is fixed to have two relays for each pressure channel.

#### **Parameters**

- adapter pyvisa resource name of the instrument or adapter instance
- **name** (*string*) The name of the instrument.
- address device address included in every message to the instrument (default=253)
- **kwargs** Any valid key-word argument for Instrument

ch\_1

#### Channel

IonGaugeAndPressureChannel

ch\_2

### Channel

PressureChannel

 $ch_3$ 

#### Channel

IonGaugeAndPressureChannel

ch\_4

### Channel

PressureChannel

 $ch_5$ 

### Channel

IonGaugeAndPressureChannel

```
ch_6
       Channel
           PressureChannel
relay_1
       Channel
           Relay
relay_10
       Channel
           Relay
relay_11
       Channel
           Relay
relay_12
       Channel
           Relay
relay_2
       Channel
           Relay
relay_3
       Channel
           Relay
relay_4
       Channel
           Relay
relay_5
       Channel
           Relay
relay_6
       Channel
           Relay
relay_7
       Channel
           Relay
relay_8
       Channel
```

7.37. MKS Instruments

Relay

### relay\_9

#### Channel

Relay

### property all\_pressures

Get pressures on all channels in selected units

### property combined\_pressure1

Get pressure on channel 1 and its combination sensor

#### property combined\_pressure2

Get pressure on channel 2 and its combination sensor

### property serial

Get the serial number of the instrument

### property unit

Control pressure unit used for all pressure readings from the instrument.

Allowed units are Unit. Torr, Unit.mbar, Unit.Pa, Unit.uHg.

### class pymeasure.instruments.mksinst.mks937b.IonGaugeAndPressureChannel(parent, id)

Bases: PressureChannel

Channel having both a pressure and an ion gauge sensor

### property ion\_gauge\_status

Get ion gauge status of the channel.

### class pymeasure.instruments.mksinst.mks937b.PressureChannel(parent, id)

Bases: Channel

# property power\_enabled

Control power status of the channel. (bool)

#### property pressure

Get the pressure on the channel in units selected on the device

# class pymeasure.instruments.mksinst.mks937b.Relay(parent, id)

Bases: RelayChannel

# property enabled

Control the relay function or disable the setpoint relay.

Possible values are True/False to enable/disable pressure based activation of the relay and 'SET' to permanently energize the relay.

### 7.37.3 MKS Instruments 974B Vacuum Pressure Transducer

Bases: MKSInstrument

MKS 974B vacuum pressure transducer

Connection to the device is made through an RS232/RS485 serial connection.

The 974B pressure transducer is a pressure gauge combining a cold cathode ionization current measurement, a Pirani sensor, and a Piezo sensor to cover a large pressure range. It optionally includes up to three mechanical relays which can be energized based on the measured pressure value.

#### **Parameters**

- adapter pyvisa resource name of the instrument or adapter instance
- **name** (*string*) The name of the instrument.
- address device address included in every message to the instrument (default=253)
- **kwargs** Any valid key-word argument for Instrument

### relay\_1

#### Channel

Relay

### relay\_2

### Channel

Relay

#### relay\_3

#### Channel

Relay

### property coldcathode\_pressure

Get Cold Cathode sensor pressure

### property device\_type

Get the device type

### property firmware\_version

Get the firmware version of the instrument

### property hardware\_version

Get the hardware version of the instrument

# id()

Get the identification of the instrument.

# property manufacturer

Get the manufacturer name

### property model

Get the transducer model number

### property operation\_hours

Get the operation hours of the instrument

### property part\_number

Get the transducer part number

### property piezo\_pressure

Get Piezo differential sensor pressure

### property pirani\_pressure

Get MicroPirani sensor pressure

7.37. MKS Instruments 475

#### property pressure

Get combined pressure reading

### property serial\_number

Get the serial number of the instrument

#### property status

Get transducer status

#### property switch\_enabled

Control the user switch to prevent accidental execution of adjustments

#### property temperature

Get the MicroPirani sensor temperature

### property unit

Control pressure unit used for all pressure readings from the instrument.

Allowed units are Unit.Torr, Unit.mbar, Unit.Pa.

### property user\_tag

Control the user programmable tag

**class** pymeasure.instruments.mksinst.mks974b.**Relay**(parent, id)

Bases: RelayChannel

### property enabled

Control the assigned input channel or disable the setpoint relay.

# 7.38 Newport

This section contains specific documentation on the Newport instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.38.1 ESP 300 Motion Controller

Bases: SCPIUnknownMixin, Instrument

Represents the Newport ESP 300 Motion Controller and provides a high-level for interacting with the instrument.

By default this instrument is constructed with x, y, and phi attributes that represent axes 1, 2, and 3. Custom implementations can overwrite this depending on the available axes. Axes are controlled through an *Axis* class.

### property axes

Get a list of the Axis objects that are present.

# clear\_errors()

Clears the error messages by checking until a 0 code is received.

### disable()

Disables all of the axes associated with this controller.

### enable()

Enables all of the axes associated with this controller.

### property error

Get an error code from the motion controller.

#### property errors

Get a list of error Exceptions that can be later raised, or used to diagnose the situation.

#### shutdown()

Shuts down the controller by disabling all of the axes.

#### **class** pymeasure.instruments.newport.esp300.**Axis**(axis, controller)

Bases: object

Represents an axis of the Newport ESP300 Motor Controller, which can have independent parameters from the other axes.

### define\_position(position)

Overwrites the value of the current position with the given value.

### disable()

Disables motion for the axis.

#### enable()

Enables motion for the axis.

#### property enabled

Returns a boolean value that is True if the motion for this axis is enabled.

### home(type=1)

Drives the axis to the home position, which may be the negative hardware limit for some actuators (e.g. LTA-HS). type can take integer values from 0 to 6.

### property left\_limit

A floating point property that controls the left software limit of the axis.

### property motion\_done

Returns a boolean that is True if the motion is finished.

### property position

A floating point property that controls the position of the axis. The units are defined based on the actuator. Use the  $wait\_for\_stop()$  method to ensure the position is stable.

#### property right\_limit

A floating point property that controls the right software limit of the axis.

### property units

A string property that controls the displacement units of the axis, which can take values of: encoder count, motor step, millimeter, micrometer, inches, milli-inches, micro-inches, degree, gradient, radian, milliradian, and microradian.

### wait\_for\_stop(delay=0, interval=0.05)

Blocks the program until the motion is completed. A further delay can be specified in seconds.

### zero()

Resets the axis position to be zero at the current poisiton.

7.38. Newport 477

```
class pymeasure.instruments.newport.esp300.AxisError(code)

Bases: Exception

Raised when a particular axis causes an error for the Newport ESP300.

class pymeasure.instruments.newport.esp300.GeneralError(code)

Bases: Exception

Raised when the Newport ESP300 has a general error.
```

# 7.39 National Instruments

This section contains specific documentation on the National Instruments instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.39.1 NI Virtual Bench

#### **General Information**

The armstrap/pyvirtualbench Python wrapper for the VirtualBench C-API is required. This Instrument driver only interfaces the pyvirtualbench Python wrapper.

### **Examples**

To be documented. Check the examples in the pyvirtualbench repository to get an idea.

Simple Example to switch digital lines of the DIO module.

```
from pymeasure.instruments.ni import VirtualBench

vb = VirtualBench(device_name='VB8012-3057E1C')
line = 'dig/2'  # may be list of lines

# initialize DIO module -> available via vb.dio
vb.acquire_digital_input_output(line, reset=False)

vb.dio.write(self.line, {True})
sleep(1000)
vb.dio.write(self.line, {False})

vb.shutdown()
```

### **Instrument Class**

```
class pymeasure.instruments.ni.virtualbench.VirtualBench(device_name=", name='VirtualBench')
Bases: object
```

Represents National Instruments Virtual Bench main frame.

Subclasses implement the functionalities of the different modules:

- Mixed-Signal-Oscilloscope (MSO)
- Digital Input Output (DIO)

- Function Generator (FGEN)
- Power Supply (PS)
- Serial Peripheral Interface (SPI) -> not implemented for pymeasure yet
- Inter Integrated Circuit (I2C) -> not implemented for pymeasure yet

For every module exist methods to save/load the configuration to file. These methods are not wrapped so far, checkout the pyvirtualbench file.

All calibration methods and classes are not wrapped so far, since these are not required on a very regular basis. Also the connections via network are not yet implemented. Check the pyvirtualbench file, if you need the functionality.

#### **Parameters**

- **device\_name** (str) Full unique device name
- name (str) Name for display in pymeasure

# class DigitalInputOutput(virtualbench, lines, reset, vb\_name=")

Bases: VirtualBenchInstrument

Represents Digital Input Output (DIO) Module of Virtual Bench device. Allows to read/write digital channels and/or set channels to export the start signal of FGEN module or trigger of MSO module.

```
export_signal(line, digitalSignalSource)
```

Exports a signal to the specified line.

#### **Parameters**

- **line** (*str*) Line string
- digitalSignalSource(int) 0 for FGEN start or 1 for MSO trigger

### query\_export\_signal(line)

Indicates the signal being exported on the specified line.

### **Parameters**

**line** (str) – Line string

#### Returns

Exported signal (FGEN start or MSO trigger)

#### Return type

enum

# query\_line\_configuration()

Indicates the current line configurations. Tristate Lines, Static Lines, and Export Lines contain commaseparated range\_data and/or colon-delimited lists of all acquired lines

#### read(lines)

Reads the current state of the specified lines.

### **Parameters**

**lines** (*str*) – Line string, requires full name specification e.g. 'VB8012-xxxxxx/dig/0:7' since instrument\_handle is not required (only library\_handle)

#### **Returns**

List of line states (HIGH/LOW)

#### Return type

list

### reset\_instrument()

Resets the session configuration to default values, and resets the device and driver software to a known state.

#### shutdown()

Removes the session and deallocates any resources acquired during the session. If output is enabled on any channels, they remain in their current state.

#### tristate\_lines(lines)

Sets all specified lines to a high-impedance state. (Default)

validate\_lines(lines, return\_single\_lines=False, validate\_init=False)

Validate lines string Allowed patterns (case sensitive):

- 'VBxxxx-xxxxxxx/dig/0:7'
- 'VBxxxx-xxxxxx/diq/0'
- 'dig/0'
- 'VBxxxx-xxxxxxx/trig'
- 'trig'

Allowed Line Numbers: 0-7 or trig

#### **Parameters**

- **lines** (*str*) Line string to test
- return\_single\_lines (bool, optional) Return list of line numbers as well, defaults to False
- validate\_init (bool, optional) Check if lines are initialized (in self. \_line\_numbers), defaults to False

#### **Returns**

Line string, optional list of single line numbers

### Return type

str, optional (str, list)

write(lines, data)

Writes data to the specified lines.

### **Parameters**

- lines (str) Line string
- data (list or tuple) List of data, (True = High, False = Low)

#### class DigitalMultimeter(virtualbench, reset, vb\_name=")

Bases: VirtualBenchInstrument

Represents Digital Multimeter (DMM) Module of Virtual Bench device. Allows to measure either DC/AC voltage or current, Resistance or Diodes.

```
configure_ac_current(auto range terminal)
```

Configure auto rage terminal for AC current measurement

#### **Parameters**

auto\_range\_terminal - Terminal to perform auto ranging ('LOW' or 'HIGH')

#### configure\_dc\_current(auto\_range\_terminal)

Configure auto rage terminal for DC current measurement

#### **Parameters**

auto\_range\_terminal - Terminal to perform auto ranging ('LOW' or 'HIGH')

### configure\_dc\_voltage(dmm\_input\_resistance)

Configure DC voltage input resistance

### **Parameters**

 $\label{local_dmm_input_resistance} \mbox{ (int or str)} - \mbox{ Input resistance ('TEN_MEGA_OHM') or 'TEN_GIGA_OHM')}$ 

```
configure_measurement(dmm_function, auto_range=True, manual_range=1.0)
    Configure Instrument to take a DMM measurement
        Parameters
          • name (dmm_function:DMM function index or) -
            - 'DC_VOLTS', 'AC_VOLTS'
            - 'DC_CURRENT', 'AC_CURRENT'
            - 'RESISTANCE'
            - 'DIODE'
          • auto_range (bool) – Enable/Disable auto ranging
          • manual_range (float) – Manually set measurement range
query_ac_current()
    Indicates auto range terminal for AC current measurement
query_dc_current()
    Indicates auto range terminal for DC current measurement
query_dc_voltage()
    Indicates input resistance setting for DC voltage measurement
query_measurement()
     Query DMM measurement settings from the instrument
         Returns
             Auto range, range data
         Return type
             (bool, float)
read()
     Read measurement value from the instrument
         Returns
             Measurement value
         Return type
             float
reset_instrument()
    Reset the DMM module to defaults
shutdown()
    Removes the session and deallocates any resources acquired during the session. If output is enabled
    on any channels, they remain in their current state.
validate_auto_range_terminal(auto_range_terminal)
     Check value for choosing the auto range terminal for DC current measurement
         Parameters
             auto_range_terminal (int or str) - Terminal to perform auto ranging ('LOW' or
             'HIGH')
         Returns
             Auto range terminal to pass to the instrument
         Return type
             int
```

### validate\_dmm\_function(dmm\_function)

Check if DMM function dmm\_function exists

#### **Parameters**

**dmm\_function** (*int or str*) – DMM function index or name:

- 'DC\_VOLTS', 'AC\_VOLTS'
- 'DC\_CURRENT', 'AC\_CURRENT'
- 'RESISTANCE'
- 'DIODE'

#### **Returns**

DMM function index to pass to the instrument

#### **Return type**

int

### static validate\_range(dmm\_function, range)

Checks if range is valid for the chosen dmm\_function

#### **Parameters**

- dmm\_function (int) DMM Function
- range (int or float) Range value, e.g. maximum value to measure

### Returns

Range value to pass to instrument

### Return type

int

### class FunctionGenerator(virtualbench, reset, vb\_name=")

Bases: VirtualBenchInstrument

Represents Function Generator (FGEN) Module of Virtual Bench device.

### configure\_arbitrary\_waveform(waveform, sample\_period)

Configures the instrument to output a waveform. The waveform is output either after the end of the current waveform if output is enabled, or immediately after output is enabled.

#### **Parameters**

- waveform (list) Waveform as list of values
- **sample\_period** (*float*) Time between two waveform points (maximum of 125MS/s, which equals 80ns)

### configure\_arbitrary\_waveform\_gain\_and\_offset(gain, dc\_offset)

Configures the instrument to output an arbitrary waveform with a specified gain and offset value. The waveform is output either after the end of the current waveform if output is enabled, or immediately after output is enabled.

#### **Parameters**

- gain (float) Gain, multiplier of waveform values
- dc\_offset (float) DC offset in volts

### configure\_standard\_waveform(waveform\_function, amplitude, dc\_offset, frequency, duty\_cycle)

Configures the instrument to output a standard waveform. Check instrument manual for maximum ratings which depend on load.

#### **Parameters**

- waveform\_function (int or str) Waveform function ("SINE", "SQUARE", "TRIANGLE/RAMP", "DC")
- amplitude (float) Amplitude in volts
- dc\_offset (float) DC offset in volts
- **frequency** (*float*) Frequency in Hz
- duty\_cycle (int) Duty cycle in %

### property filter

Enables or disables the filter on the instrument.

#### **Parameters**

enable\_filter (bool) - Enable/Disable filter

### query\_arbitrary\_waveform()

Returns the samples per second for arbitrary waveform generation.

#### Returns

Samples per second

### Return type

int

### query\_arbitrary\_waveform\_gain\_and\_offset()

Returns the settings for arbitrary waveform generation that includes gain and offset settings.

#### Returns

Gain, DC offset

### Return type

(float, float)

### query\_generation\_status()

Returns the status of waveform generation on the instrument.

# Returns

Status

#### Return type

enum

# query\_standard\_waveform()

Returns the settings for a standard waveform generation.

#### **Returns**

Waveform function, amplitude, dc\_offset, frequency, duty\_cycle

#### Return type

(enum, float, float, float, int)

### query\_waveform\_mode()

Indicates whether the waveform output by the instrument is a standard or arbitrary waveform.

#### Returns

Waveform mode

### **Return type**

enum

# reset\_instrument()

Resets the session configuration to default values, and resets the device and driver software to a known state.

#### run()

Transitions the session from the Stopped state to the Running state.

#### self\_calibrate()

Performs offset nulling calibration on the device. You must run FGEN Initialize prior to running this method.

#### shutdown()

Removes the session and deallocates any resources acquired during the session. If output is enabled on any channels, they remain in their current state.

#### stop()

Transitions the acquisition from either the Triggered or Running state to the Stopped state.

### class MixedSignalOscilloscope(virtualbench, reset, vb\_name=")

Bases: VirtualBenchInstrument

Represents Mixed Signal Oscilloscope (MSO) Module of Virtual Bench device. Allows to measure oscilloscope data from analog and digital channels.

Methods from pyvirtualbench not implemented in pymeasure yet:

- enable\_digital\_channels
- configure\_digital\_threshold
- configure\_advanced\_digital\_timing
- configure\_state\_mode
- configure\_digital\_edge\_trigger
- configure\_digital\_pattern\_trigger
- configure\_digital\_glitch\_trigger
- configure\_digital\_pulse\_width\_trigger
- query\_digital\_channel
- query\_enabled\_digital\_channels
- query\_digital\_threshold
- query\_advanced\_digital\_timing
- query\_state\_mode
- query\_digital\_edge\_trigger
- query\_digital\_pattern\_trigger
- query\_digital\_glitch\_trigger
- query\_digital\_pulse\_width\_trigger

• read\_digital\_u64

#### auto\_setup()

Automatically configure the instrument

Configure analog measurement channel

### **Parameters**

- **channel** (*str*) Channel string
- enable\_channel (bool) Enable/Disable channel
- **vertical\_range** (*float*) Vertical measurement range (0V 20V), the instrument discretizes to these ranges: [20, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05] which are 5x the values shown in the native UI.
- **vertical\_offset** (*float*) Vertical offset to correct for (inverted compared to VB native UI, -20V +20V, resolution 0.1mV)
- probe\_attenuation (int or str) Probe attenuation ('ATTENUATION\_10X' or 'ATTENUATION\_1X')
- vertical\_coupling (int or str) Vertical coupling ('AC' or 'DC')

 ${\bf configure\_analog\_channel\_characteristics} ({\it channel}, {\it input\_impedance}, {\it bandwidth\_limit})$ 

Configure electrical characteristics of the specified channel

#### **Parameters**

- **channel** (*str*) Channel string
- input\_impedance (int or str) Input Impedance ('ONE\_MEGA\_OHM' or 'FIFTY\_OHMS')
- bandwidth\_limit (int) Bandwidth limit (100MHz or 20MHz)

Configures a trigger to activate on the specified source when the analog edge reaches the specified levels.

### **Parameters**

- **trigger\_source** (*str*) Channel string
- trigger\_slope (int or str) Trigger slope ('RISING', 'FALLING' or 'EITHER')
- trigger\_level (float) Trigger level
- trigger\_hysteresis (float) Trigger hysteresis
- **trigger\_instance** (*int or str*) Trigger instance

Configures a trigger to activate on the specified source when the analog edge reaches the specified levels within a specified window of time.

#### **Parameters**

- **trigger\_source** (*str*) Channel string
- trigger\_polarity(int or str) Trigger slope('POSITIVE' or 'NEGATIVE')
- trigger\_level (float) Trigger level
- comparison\_mode (int or str) Mode of compariosn (
  'GREATER\_THAN\_UPPER\_LIMIT',
  'INSIDE\_LIMITS' or 'OUTSIDE\_LIMITS')
  'LESS\_THAN\_LOWER\_LIMIT',
- **lower\_limit** (*float*) Lower limit
- upper\_limit (float) Upper limit
- trigger\_instance (int or str) Trigger instance

### configure\_immediate\_trigger()

Configures a trigger to immediately activate on the specified channels after the pretrigger time has expired.

**configure\_timing**(sample\_rate, acquisition\_time, pretrigger\_time, sampling\_mode)

Configure timing settings of the MSO

#### **Parameters**

- sample\_rate (int) Sample rate (15.26kS 1GS)
- acquisition\_time (float) Acquisition time (1ns 68.711s)
- **pretrigger\_time** (*float*) Pretrigger time (0s 10s)
- sampling\_mode Sampling mode ('SAMPLE' or 'PEAK\_DETECT')

### configure\_trigger\_delay(trigger\_delay)

Configures the amount of time to wait after a trigger condition is met before triggering.

# param float trigger\_delay

Trigger delay (0s - 17.1799s)

#### force\_trigger()

Causes a software-timed trigger to occur after the pretrigger time has expired.

### query\_acquisition\_status()

Returns the status of a completed or ongoing acquisition.

### query\_analog\_channel(channel)

Indicates the vertical configuration of the specified channel.

#### Returns

Channel enabled, vertical range, vertical offset, probe attenuation, vertical coupling

### **Return type**

(bool, float, float, enum, enum)

### query\_analog\_channel\_characteristics(channel)

Indicates the properties that control the electrical characteristics of the specified channel. This method returns an error if too much power is applied to the channel.

#### return

Input impedance, bandwidth limit

#### rtype

(enum, float)

### query\_analog\_edge\_trigger(trigger\_instance)

Indicates the analog edge trigger configuration of the specified instance.

#### Returns

Trigger source, trigger slope, trigger level, trigger hysteresis

#### Return type

(str, enum, float, float)

#### query\_analog\_pulse\_width\_trigger(trigger instance)

Indicates the analog pulse width trigger configuration of the specified instance.

#### **Returns**

Trigger source, trigger polarity, trigger level, comparison mode, lower limit, upper limit

### Return type

(str, enum, float, enum, float, float)

### query\_enabled\_analog\_channels()

Returns String of enabled analog channels.

#### Returns

Enabled analog channels

### Return type

str

### query\_timing()

Indicates the timing configuration of the MSO. Call directly before measurement to read the actual timing configuration and write it to the corresponding class variables. Necessary to interpret the measurement data, since it contains no time information.

### Returns

Sample rate, acquisition time, pretrigger time, sampling mode

### Return type

(float, float, float, enum)

### query\_trigger\_delay()

Indicates the trigger delay setting of the MSO.

### Returns

Trigger delay

# Return type

float

### query\_trigger\_type(trigger\_instance)

Indicates the trigger type of the specified instance.

### **Parameters**

```
trigger_instance - Trigger instance ('A' or 'B')
```

#### **Returns**

Trigger type

### Return type

str

### read\_analog\_digital\_dataframe()

Transfers data from the instrument and returns a pandas dataframe of the analog measurement data, including time coordinates

#### **Returns**

Dataframe with time and measurement data

### **Return type**

pd.DataFrame

### read\_analog\_digital\_u64()

Transfers data from the instrument as long as the acquisition state is Acquisition Complete. If the state is either Running or Triggered, this method will wait until the state transitions to Acquisition Complete. If the state is Stopped, this method returns an error.

#### Returns

Analog data out, analog data stride, analog t0, digital data out, digital timestamps out, digital t0, trigger timestamp, trigger reason

#### **Return type**

(list, int, pyvb.Timestamp, list, list, pyvb.Timestamp, pyvb.Timestamp, enum)

#### reset\_instrument()

Resets the session configuration to default values, and resets the device and driver software to a known state.

```
run(autoTrigger=True)
```

Transitions the acquisition from the Stopped state to the Running state. If the current state is Triggered, the acquisition is first transitioned to the Stopped state before transitioning to the Running state. This method returns an error if too much power is applied to any enabled channel.

#### **Parameters**

**autoTrigger** (bool) – Enable/Disable auto triggering

### shutdown()

Removes the session and deallocates any resources acquired during the session. If output is enabled on any channels, they remain in their current state.

#### stop()

Transitions the acquisition from either the Triggered or Running state to the Stopped state.

#### validate\_channel(channel)

Check if channel is a correct specification

#### **Parameters**

**channel** (str) – Channel string

#### Returns

Channel string

#### Return type

str

# static validate\_trigger\_instance(trigger\_instance)

Check if trigger\_instance is a valid choice

#### **Parameters**

trigger\_instance (int or str) - Trigger instance ('A' or 'B')

#### Returns

Trigger instance

# Return type

int

#### class PowerSupply(virtualbench, reset, vb\_name=")

Bases: VirtualBenchInstrument

Represents Power Supply (PS) Module of Virtual Bench device

#### configure\_current\_output(channel, current\_level, voltage\_limit)

Configures a current output on the specified channel. This method should be called once for every channel you want to configure to output current.

#### configure\_voltage\_output(channel, voltage\_level, current\_limit)

Configures a voltage output on the specified channel. This method should be called once for every channel you want to configure to output voltage.

### property outputs\_enabled

Enables or disables all outputs on all channels of the instrument.

#### **Parameters**

enable\_outputs (bool) - Enable/Disable outputs

### query\_current\_output(channel)

Indicates the current output settings on the specified channel.

### query\_voltage\_output(channel)

Indicates the voltage output settings on the specified channel.

### read\_output(channel)

Reads the voltage and current levels and outout mode of the specified channel.

#### reset\_instrument()

Resets the session configuration to default values, and resets the device and driver software to a known state.

#### shutdown()

Removes the session and deallocates any resources acquired during the session. If output is enabled on any channels, they remain in their current state.

### property tracking

Enables or disables tracking between the positive and negative 25V channels. If enabled, any configuration change on the positive 25V channel is mirrored to the negative 25V channel, and any writes to the negative 25V channel are ignored.

### **Parameters**

enable\_tracking (bool) - Enable/Disable tracking

#### validate\_channel(channel, current=False, voltage=False)

Check if channel string is valid and if output current/voltage are within the output ranges of the channel

#### **Parameters**

- **channel** (*str*) Channel string ("ps/+6V", "ps/+25V", "ps/-25V")
- current (bool, optional) Current output, defaults to False
- voltage (bool, optional) Voltage output, defaults to False

### Returns

channel or channel, current & voltage

### Return type

str or (str, float, float)

# acquire\_digital\_input\_output(lines, reset=False)

Establishes communication with the DIO module. This method should be called once per session.

### **Parameters**

- lines (str) Lines to acquire, reading is possible on all lines
- reset (bool, optional) Reset DIO module, defaults to False

### acquire\_digital\_multimeter(reset=False)

Establishes communication with the DMM module. This method should be called once per session.

#### **Parameters**

reset (bool, optional) – Reset the DMM module, defaults to False

#### acquire\_function\_generator(reset=False)

Establishes communication with the FGEN module. This method should be called once per session.

### **Parameters**

reset (bool, optional) - Reset the FGEN module, defaults to False

### acquire\_mixed\_signal\_oscilloscope(reset=False)

Establishes communication with the MSO module. This method should be called once per session.

#### **Parameters**

reset (bool, optional) - Reset the MSO module, defaults to False

#### acquire\_power\_supply(reset=False)

Establishes communication with the PS module. This method should be called once per session.

#### **Parameters**

reset (bool, optional) - Reset the PS module, defaults to False

# collapse\_channel\_string(names\_in)

Collapses a channel string into a comma and colon-delimited equivalent. Last element is the number of channels.

### **Parameters**

**names\_in** (*str*) – Channel string

#### **Returns**

Channel string with colon notation where possible, number of channels

#### Return type

(str, int)

### convert\_timestamp\_to\_values(timestamp)

Converts a timestamp to seconds and fractional seconds

#### **Parameters**

timestamp (pyvb. Timestamp) – VirtualBench timestamp

#### Returns

(seconds\_since\_1970, fractional seconds)

# Return type

(int, float)

# convert\_values\_to\_datetime(timestamp)

Converts timestamp to datetime object

### **Parameters**

timestamp (pyvb. Timestamp) – VirtualBench timestamp

#### **Returns**

Timestamp as DateTime object

### **Return type**

DateTime

# ${\tt convert\_values\_to\_timestamp} (seconds\_since\_1970, fractional\_seconds)$

Converts seconds and fractional seconds to a timestamp

#### **Parameters**

- seconds\_since\_1970 (int) Date/Time in seconds since 1970
- **fractional\_seconds** (*float*) Fractional seconds

#### **Returns**

VirtualBench timestamp

#### **Return type**

pyvb.Timestamp

### expand\_channel\_string(names\_in)

Expands a channel string into a comma-delimited (no colon) equivalent. Last element is the number of channels.  $\frac{\text{dig}}{0:2'} - \frac{\text{dig}}{0}$ ,  $\frac{\text{dig}}{1}$ ,  $\frac{\text{dig}}{2'}$ , 3)

#### **Parameters**

names\_in (str) - Channel string

#### Returns

Channel string with all channels separated by comma, number of channels

### Return type

(str, int)

# get\_calibration\_information()

Returns calibration information for the specified device, including the last calibration date and calibration interval.

#### Returns

Calibration date, recommended calibration interval in months, calibration interval in months

# Return type

(pyvb.Timestamp, int, int)

#### get\_library\_version()

Return the version of the VirtualBench runtime library

### shutdown()

Finalize the VirtualBench library.

# class pymeasure.instruments.ni.virtualbench.VirtualBench\_Direct(\*args: Any, \*\*kwargs: Any)

Bases: PyVirtualBench

Represents National Instruments Virtual Bench main frame. This class provides direct access to the arm-strap/pyvirtualbench Python wrapper.

# 7.40 Novanta Photonics

This section contains specific documentation on the Novanta photonics instruments that are implemented. Novanta contains also Lasers developed by Laserquantum. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.40.1 Novanta FPU60 laser power supply unit

Bases: Instrument

Represents a fpu60 power supply unit for the finesse laser series by Laserquantum, a Novanta company.

The instrument responds to every command sent.

### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

### Returns

List of error entries.

### property current

Measure the diode current in percent (float).

### disable\_emission()

Disable emission and unlock the button afterwards.

You have to press the physical button to enable emission again.

#### property emission\_enabled

Measure the emission status (bool).

### get\_operation\_times()

Get the operation times in minutes as a dictionary.

# property head\_temperature

Measure the laser head temperature in °C (float).

### property interlock\_enabled

Get the interlock enabled status (bool).

### property power

Measure current output power in Watts (float).

### property power\_setpoint

Control the output power setpoint in Watts (float).

### property psu\_temperature

Measure the power supply unit temperature in °C (float).

### property serial\_number

Get the serial number (str).

#### property shutter\_open

Control whether the shutter is open (bool).

### property software\_version

Get the software version (str).

# 7.41 Oxford Instruments

This section contains specific documentation on the Oxford Instruments instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.41.1 Oxford Instruments Base Instrument

class pymeasure.instruments.oxfordinstruments.base.OxfordInstrumentsBase(adapter,

name='OxfordInstruments Base', max\_attempts=5, \*\*kwargs)

Bases: Instrument

Base instrument for devices from Oxford Instruments.

Checks the replies from instruments for validity.

#### **Parameters**

- adapter A string, integer, or *Adapter* subclass object
- name (string) The name of the instrument. Often the model designation by default.
- max\_attempts Integer that sets how many attempts at getting a valid response to a query can be made
- \*\*kwargs In case adapter is a string or integer, additional arguments passed on to VISAAdapter (check there for details). Discarded otherwise.

### ask(command)

Write the command to the instrument and return the resulting ASCII response. Also check the validity of the response before returning it; if the response is not valid, another attempt is made at getting a valid response, until the maximum amount of attempts is reached.

#### **Parameters**

**command** – ASCII command string to be sent to the instrument

#### Returns

String ASCII response of the instrument

#### Raises

OxfordVISAError if the maximum number of attempts is surpassed without getting a valid response

# is\_valid\_response(response, command)

Check if the response received from the instrument after a command is valid and understood by the instrument.

### **Parameters**

• response – String ASCII response of the device

• **command** – command used in the initial query

#### Returns

True if the response is valid and the response indicates the instrument recognised the command

#### write(command)

Write command to instrument and check whether the reply indicates that the given command was not understood. The devices from Oxford Instruments reply with '?xxx' to a command 'xxx' if this command is not known, and replies with 'x' if the command is understood. If the command starts with an "\$" the instrument will not reply at all; hence in that case there will be done no checking for a reply.

#### Raises

OxfordVISAError if the instrument does not recognise the supplied command or if the response of the instrument is not understood

class pymeasure.instruments.oxfordinstruments.base.OxfordVISAError

Bases: Exception

# 7.41.2 Oxford Instruments Intelligent Temperature Controller 503

Bases: OxfordInstrumentsBase

Represents the Oxford Intelligent Temperature Controller 503.

```
itc = ITC503("GPIB::24")  # Default channel for the ITC503

itc.control_mode = "RU"  # Set the control mode to remote
itc.heater_gas_mode = "AUTO"  # Turn on auto heater and flow
itc.auto_pid = True  # Turn on auto-pid

print(itc.temperature_setpoint) # Print the current set-point
itc.temperature_setpoint = 300  # Change the set-point to 300 K
itc.wait_for_temperature()  # Wait for the temperature to stabilize
print(itc.temperature_1)  # Print the temperature at sensor 1
```

**class FLOW\_CONTROL\_STATUS**(value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntFlag

IntFlag class for decoding the flow control status. Contains the following flags:

bit	flag	meaning
4	HEATER_ERROR_SIGN	Sign of heater-error; True means negative
3	TEMPERATURE_ERROR_SIGN	Sign of temperature-error; True means negative
2	SLOW_VALVE_ACTION	Slow valve action occurring
1	COOLDOWN_TERMINATION	Cooldown-termination occurring
0	FAST_COOLDOWN	Fast-cooldown occurring

#### property auto\_pid

A boolean property that sets the Auto-PID mode on (True) or off (False).

### property auto\_pid\_table

A property that controls values in the auto-pid table. Relies on  $ITC503.x\_pointer$  and  $ITC503.y\_pointer$  (or ITC503.pointer) to point at the location in the table that is to be set or read.

The x-pointer selects the table entry (1 to 16); the y-pointer selects the parameter:

y-pointer	parameter
1	upper temperature limit
2	proportional band
3	integral action time
4	derivative action time

### property control\_mode

A string property that sets the ITC in *local* or *remote* and *locked* or *unlocked*, locking the LOC/REM button. Allowed values are:

value	state
LL	local & locked
RL	remote & locked
LU	local & unlocked
RU	remote & unlocked

### property derivative\_action\_time

A floating point property that controls the derivative action time for the PID controller in minutes. Can be set if the PID controller is in manual mode. Valid values are 0 [min.] to 273 [min.].

### property front\_panel\_display

A string property that controls what value is displayed on the front panel of the ITC. Valid values are: 'temperature setpoint', 'temperature 1', 'temperature 2', 'temperature 3', 'temperature error', 'heater', 'heater voltage', 'gasflow', 'proportional band', 'integral action time', 'derivative action time', 'channel 1 freq/4', 'channel 2 freq/4', 'channel 3 freq/4'.

### property gasflow

A floating point property that controls gas flow when in manual mode. The value is expressed as a percentage of the maximum gas flow. Valid values are in range 0 [off] to 99.9 [%].

### property gasflow\_configuration\_parameter

A property that controls the gas flow configuration parameters. Relies on the *ITC503.x\_pointer* to select which parameter is set or read:

x-pointer	parameter
1	valve gearing
2	target table & features configuration
3	gas flow scaling
4	temperature error sensitivity
5	heater voltage error sensitivity
6	minimum gas valve in auto

#### property gasflow\_control\_status

A property that reads the gas-flow control status. Returns the status in the form of a ITC503. FLOW\_CONTROL\_STATUS IntFlag.

#### property heater

A floating point property that represents the heater output power as a percentage of the maximum voltage. Can be set if the heater is in manual mode. Valid values are in range 0 [off] to 99.9 [%].

#### property heater\_gas\_mode

A string property that sets the heater and gas flow control to *auto* or *manual*. Allowed values are:

value	state
MANUAL	heater & gas manual
AM	heater auto, gas manual
MA	heater manual, gas auto
AUTO	heater & gas auto

### property heater\_voltage

A floating point property that represents the heater output power in volts. For controlling the heater, use the *ITC503.heater* property.

### property integral\_action\_time

A floating point property that controls the integral action time for the PID controller in minutes. Can be set if the PID controller is in manual mode. Valid values are 0 [min.] to 140 [min.].

# property pointer

A tuple property to set pointers into tables for loading and examining values in the table, of format (x, y). The significance and valid values for the pointer depends on what property is to be read or set. The value for x and y can be in the range 0 to 128.

#### program\_sweep(temperatures, sweep\_time, hold\_time, steps=None)

Program a temperature sweep in the controller. Stops any running sweep. After programming the sweep, it can be started using OxfordITC503.sweep\_status = 1.

#### **Parameters**

- **temperatures** An array containing the temperatures for the sweep
- **sweep\_time** The time (or an array of times) to sweep to a set-point in minutes (between 0 and 1339.9).
- **hold\_time** The time (or an array of times) to hold at a set-point in minutes (between 0 and 1339.9).
- **steps** The number of steps in the sweep, if given, the temperatures, sweep\_time and hold\_time will be interpolated into (approximately) equal segments

#### property proportional\_band

A floating point property that controls the proportional band for the PID controller in Kelvin. Can be set if the PID controller is in manual mode. Valid values are 0 [K] to 1677.7 [K].

### property sweep\_status

An integer property that sets the sweep status. Values are:

value	meaning
0	Sweep not running
1	Start sweep / sweeping to first set-point
2P - 1	Sweeping to set-point P
2P	Holding at set-point P

### property sweep\_table

A property that controls values in the sweep table. Relies on  $ITC503.x\_pointer$  and  $ITC503.y\_pointer$  (or ITC503.pointer) to point at the location in the table that is to be set or read.

The x-pointer selects the step of the sweep (1 to 16); the y-pointer selects the parameter:

y-pointer	parameter
1	set-point temperature
2	sweep-time to set-point
3	hold-time at set-point

### property target\_voltage

A float property that reads the current heater target voltage with which the actual heater voltage is being compared. Only valid if gas-flow in auto mode.

#### property target\_voltage\_table

A property that controls values in the target heater voltage table. Relies on the *ITC503.x\_pointer* to select the entry in the table that is to be set or read (1 to 64).

#### property temperature\_1

Reads the temperature of the sensor 1 in Kelvin.

### property temperature\_2

Reads the temperature of the sensor 2 in Kelvin.

### property temperature\_3

Reads the temperature of the sensor 3 in Kelvin.

# property temperature\_error

Reads the difference between the set-point and the measured temperature in Kelvin. Positive when set-point is larger than measured.

### property temperature\_setpoint

A floating point property that controls the temperature set-point of the ITC in kelvin. (dynamic)

#### property valve\_scaling

A float property that reads the valve scaling parameter. Only valid if gas-flow in auto mode.

### property version

A string property that returns the version of the IPS.

Wait for the ITC to reach the set-point temperature.

#### **Parameters**

- **error** The maximum error in Kelvin under which the temperature is considered at set-point
- **timeout** The maximum time the waiting is allowed to take. If timeout is exceeded, a TimeoutError is raised. If timeout is None, no timeout will be used.
- **check\_interval** The time between temperature queries to the ITC.
- **stability\_interval** The time over which the temperature\_error is to be below error to be considered stable.
- **thermalize\_interval** The time to wait after stabilizing for the system to thermalize.
- **should\_stop** Optional function (returning a bool) to allow the waiting to be stopped before its end.

### wipe\_sweep\_table()

Wipe the currently programmed sweep table.

#### property x\_pointer

An integer property to set pointers into tables for loading and examining values in the table. The significance and valid values for the pointer depends on what property is to be read or set.

### property y\_pointer

An integer property to set pointers into tables for loading and examining values in the table. The significance and valid values for the pointer depends on what property is to be read or set.

# 7.41.3 Oxford Instruments Intelligent Power Supply 120-10 for superconducting magnets

Bases: OxfordInstrumentsBase

Represents the Oxford Superconducting Magnet Power Supply IPS 120-10.

```
ips = IPS120_10("GPIB::25") # Default channel for the IPS
ips.enable_control()
                             # Enables the power supply and remote control
                             # Train the magnet after it has been cooled-down
ips.train_magnet([
    (11.8, 1.0),
    (13.9, 0.4),
    (14.9, 0.2),
    (16.0, 0.1),
1)
                            # Bring the magnet to 12 T. The switch heater will
ips.set_field(12)
                            # be turned off when the field is reached and the
                            # current is ramped back to 0 (i.e. persistent mode).
print(self.field)
                            # Print the current field (whether in persistent or
```

(continues on next page)

(continued from previous page)

#### **Parameters**

- **clear\_buffer** A boolean property that controls whether the instrument buffer is clear upon initialisation.
- **switch\_heater\_heating\_delay** The time in seconds (default is 20s) to wait after the switch-heater is turned on before the heater is expected to be heated.
- **switch\_heater\_cooling\_delay** The time in seconds (default is 20s) to wait after the switch-heater is turned off before the heater is expected to be cooled down.
- **field\_range** A numeric value or a tuple of two values to indicate the lowest and highest allowed magnetic fields. If a numeric value is provided the range is expected to be from -field\_range to +field\_range. The default range is -7 to +7 Tesla.

### property activity

A string property that controls the activity of the IPS. Valid values are "hold", "to setpoint", "to zero" and "clamp"

### property control\_mode

A string property that sets the IPS in *local* or *remote* and *locked* or *unlocked*, locking the LOC/REM button. Allowed values are:

value	state
LL	local & locked
RL	remote & locked
LU	local & unlocked
RU	remote & unlocked

### property current\_measured

A floating point property that returns the measured magnet current of the IPS in amps. (dynamic)

#### property current\_setpoint

A floating point property that controls the magnet current set-point of the IPS in ampere. (dynamic)

### property demand\_current

A floating point property that returns the demand magnet current of the IPS in amps. (dynamic)

#### property demand\_field

A floating point property that returns the demand magnetic field of the IPS in Tesla. (dynamic)

### disable\_control()

Disable active control of the IPS (if at 0T) by turning off the switch heater, clamping the output and setting control to local. Raise a *MagnetError* if field not at 0T.

#### disable\_persistent\_mode()

Disable the persistent magnetic field mode. Raise a MagnetError if the magnet is not at rest.

#### enable\_control()

Enable active control of the IPS by setting control to remote and turning off the clamp.

### enable\_persistent\_mode()

Enable the persistent magnetic field mode. Raise a MagnetError if the magnet is not at rest.

### property field

Property that returns the current magnetic field value in Tesla.

### property field\_setpoint

A floating point property that controls the magnetic field set-point of the IPS in Tesla. (dynamic)

### property persistent\_field

A floating point property that returns the persistent magnetic field of the IPS in Tesla. (dynamic)

### **set\_field**(*field*, *sweep\_rate=None*, *persistent\_mode\_control=True*)

Change the applied magnetic field to a new specified magnitude. If allowed (via *persistent\_mode\_control*) the persistent mode will be turned off if needed and turned on when the magnetic field is reached. When the new field set-point is 0, the set-point of the instrument will not be changed but rather the *to zero* functionality will be used. Also, the persistent mode will not turned on upon reaching the 0T field in this case.

#### **Parameters**

- **field** The new set-point for the magnetic field in Tesla.
- sweep\_rate A numeric value that controls the rate with which to change the magnetic field in Tesla/minute.
- **persistent\_mode\_control** A boolean that controls whether the persistent mode may be turned off (if needed before sweeping) and on (when the field is reached); if set to False but the system is in persistent mode, a *MagnetError* will be raised and the magnetic field will not be changed.

#### property sweep\_rate

A floating point property that controls the sweep-rate of the IPS in Tesla/minute. (dynamic)

### property sweep\_status

A string property that returns the current sweeping mode of the IPS.

#### property switch\_heater\_enabled

A boolean property that controls whether the switch heater is enabled or not. When the switch heater is enabled (True), the switch is closed and the switch is open and the current in the magnet can be controlled; when the switch heater is disabled (False) the switch is closed and the current in the magnet cannot be controlled.

When turning on the switch heater with True, the switch heater is only activated if the current of the power supply matches the last recorded current in the magnet.

### Warning

These checks can be omitted by using "Force" in stead of True. Caution: Not performing these checks can cause serious damage to both the power supply and the magnet.

After turning on the switch heater it is necessary to wait several seconds for the switch the respond.

Raises a *SwitchHeaterError* if the system reports a 'heater fault' or if no switch is fitted on the system upon getting the status.

### property switch\_heater\_status

An integer property that returns the switch heater status of the IPS. Use the <code>switch\_heater\_enabled</code> property for controlling and reading the switch heater. When using this property, the user is referred to the IPS120-10 manual for the meaning of the integer values.

### train\_magnet(training\_scheme)

Train the magnet after cooling down. Afterwards, set the field back to 0 tesla (at last-used ramp-rate).

#### **Parameters**

**training\_scheme** – The training scheme as a list of tuples; each tuple should consist of a (field [T], ramp-rate [T/min]) pair.

### property version

A string property that returns the version of the IPS.

wait\_for\_idle(delay=1, max\_wait\_time=None, should\_stop=<function IPS120\_10.<lambda>>)
Wait until the system is at rest (i.e. current of field not ramping).

#### **Parameters**

- **delay** Time in seconds between each query into the state of the instrument.
- max\_wait\_time Maximum time in seconds to wait before is at rest. If the system is not at rest within this time a TimeoutError is raised. None is interpreted as no maximum time.
- **should\_stop** A function that returns True when this function should return early.

class pymeasure.instruments.oxfordinstruments.ips120\_10.MagnetError

Bases: ValueError

Exception that is raised for issues regarding the state of the magnet or power supply.

class pymeasure.instruments.oxfordinstruments.ips120\_10.SwitchHeaterError

Bases: ValueError

Exception that is raised for issues regarding the state of the superconducting switch.

### 7.41.4 Oxford Instruments Mercury Intelligent Temperature Controller

Represents the Oxford Instruments Mercury iTC (Intelligent Temperature Controller) and provides a high-level interface for interacting with the instrument. Note thath the GPIB implementation on this instrument is not fully SCPI-compliant, with only the \*IDN? command supported. The Mercury iTC can be configured to respond to a so-called SCPI instruction set (that is, again, not actually an SCPI implementation) or to a legacy instruction set. The implementation used here is the so-called SCPI instruction set.

The iTC is expandable and supports a mixture temperature sensors, heaters, pressure sensors, cryogen level sensors and auxiliary I/O through the installation of additional, function-specific daughter boards. Currently, only temperature sensors and heaters are implemented.

By default, there is a temperature sensor channel installed with Unique Identifier (UID) 'MB1.T1' and a heater channel with UID 'MB0.H1'. These channels are created automatically at instantiation of the controller. These

can be accessed as either .TS\_MB and .HTR\_MB, or through the collections (dictionaries) .TS['MB1.T1'] and .HTR['MB0.H1'].

Other sensors and heaters can be installed on additional daughter boards (with specific UID e.g. 'DBX. T1', 'DBX.H1' etc), and can be added manually to an already instantiated MercuryiTC using . add\_temperature\_sensor(UID) and .add\_heater(UID). They must be given their correct UID as listed by the Mercury iTC itself as this is how they are addressed during communication. Additionally, they are accessed **only** through the instrument collections .TS[UID] and .HTR[UID].

### HTR\_MB

#### Channel

Heater

TS\_MB

### Channel

TemperatureSensor

```
add_heater(id, prefix='HTR', **kwargs)
```

Add a heater with unique identifier id to collection of addressable heaters: .HTR[id]

```
add_temperature_sensor(id, prefix='TS', **kwargs)
```

Add a temperature sensor with unique identifier id to collection of addressable temperature sensors: . TS[id]

### property identity

Get identity of unit.

**class** pymeasure.instruments.oxfordinstruments.mercuryitc.**TemperatureSensor**(parent, id)

Bases: Channel

A temperature sensor channel on the Oxford Instruments Mercury iTC.

Depending on sensor type (e.g. diode, NTC/PTC resistor), some properties will not be valid. Therefore, sensor configuration must be known and set a priori. Only diode sensor types are currently implemented.

Control loops are accessed via temeperature sensors, rather than heaters. Therefore loop control is kept with other temperature sensor properties.

#### check\_set\_errors()

The MercuryiTC responds to every communication (get and set). This simply reads back on setting a property.

### property control\_loop\_D

Control the derivative term, D, of the control loop.

### property control\_loop\_I

Control the integral term, I, of the control loop.

### property control\_loop\_P

Control the proportional term, P, of the control loop.

### property control\_loop\_PID\_enabled

Control whether the control-loop heater is controlled by the PID loop (True) or the manual heater percentage (False).

### property control\_loop\_heater\_percent

Control the heater percentage when output configured to manual mode (in the range 0 to 100).

### property control\_loop\_ramp\_enabled

Control whether the temperature ramping function is enabled (True) or disabled (False).

### property control\_loop\_ramp\_rate

Control the control loop ramp rate, in K/min, if enabled (in the range 0 to 100000).

### property control\_loop\_temperature\_setpoint

Control the control loop temperature setpoint, in Kelvin (in the range 0 to 2000).

### property temperature

Get the measured temperature, in Kelvin.

### property voltage

Get the sensor voltage, in Volts.

### class pymeasure.instruments.oxfordinstruments.mercuryitc.Heater(parent, id)

Bases: Channel

A heater channel on the Oxford Instruments Mercury iTC.

Various parameters of the heater output can be accessed, and some properties can be configured. PID control is accessed through an associated temperature sensor. It is assumed that the heater is already associated with a temperature sensor control loop, or otherwise set to operate in open-loop configuration.

### check\_set\_errors()

The MercuryiTC responds to every communication (get and set). This simply reads back on setting a property.

#### property current

Get the heater excitation current, in Amps.

### property max\_power

Get the provisional maximum power of the heater, in Watts.

### property power

Get the heater power dissipation, in Watts.

#### property resistance

Control the programmed heater resistance, in Ohms (float strictly in the range of 10 to 2000).

### property voltage

Get the heater excitation voltage, in Volts.

### property voltage\_limit

Control the voltage limit of the heater output, in Volts (float strictly from 0 to 40).

### 7.41.5 Oxford Instruments Power Supply 120-10 for superconducting magnets

 $\textbf{class} \ \ \textbf{pymeasure.instruments.oxfordinstruments.PS120\_10} (\textit{adapter}, \textit{name} = 'Oxford PS', **kwargs')$ 

Bases: IPS120\_10

Represents the Oxford Superconducting Magnet Power Supply PS 120-10.

```
ps = PS120_10("GPIB::25")
                            # Default channel for the IPS
                            # Enables the power supply and remote control
ps.enable_control()
                            # Train the magnet after it has been cooled-down
ps.train_magnet([
    (11.8, 1.0),
    (13.9, 0.4),
    (14.9, 0.2),
    (16.0, 0.1),
1)
ps.set_field(12)
                            # Bring the magnet to 12 T. The switch heater will
                            # be turned off when the field is reached and the
                            # current is ramped back to 0 (i.e. persistent mode).
print(self.field)
                            # Print the current field (whether in persistent or
                            # non-persistent mode)
ps.set_field(0)
                            # Bring the magnet to 0 T. The persistent mode will be
                            # turned off first (i.e. current back to set-point and
                            # switch-heater on); afterwards the switch-heater will
                            # again be turned off.
ps.disable_control()
                            # Disables the control of the supply, turns off the
                            # switch-heater and clamps the output.
```

#### **Parameters**

- **clear\_buffer** A boolean property that controls whether the instrument buffer is clear upon initialisation.
- **switch\_heater\_heating\_delay** The time in seconds (default is 20s) to wait after the switch-heater is turned on before the heater is expected to be heated.
- **switch\_heater\_cooling\_delay** The time in seconds (default is 20s) to wait after the switch-heater is turned off before the heater is expected to be cooled down.
- **field\_range** A numeric value or a tuple of two values to indicate the lowest and highest allowed magnetic fields. If a numeric value is provided the range is expected to be from -field\_range to +field\_range.

class pymeasure.instruments.oxfordinstruments.ips120\_10.MagnetError

Bases: ValueError

Exception that is raised for issues regarding the state of the magnet or power supply.

### class pymeasure.instruments.oxfordinstruments.ips120\_10.SwitchHeaterError

Bases: ValueError

Exception that is raised for issues regarding the state of the superconducting switch.

### 7.42 Parker

This section contains specific documentation on the Parker instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.42.1 Parker GV6 Servo Motor Controller

Bases: SCPIUnknownMixin, Instrument

Represents the Parker Gemini GV6 Servo Motor Controller and provides a high-level interface for interacting with the instrument

### property acceleration

Set the acceleration setpoint in revolutions per second squared.

### property angle

Control the angle in degrees based on the position and whether relative or absolute positioning is enabled,

### property angle\_error

Get the angle error in degrees based on the position error, or returns None on error

### property average\_acceleration

Set the average acceleration setpoint in revolutions per second squared.

#### disable()

Disables the motor from moving

### property echo

Set whether the echoing of all commands sent to the instrument is enabled (bool).

#### enable()

Enables the motor to move

### is\_moving()

Return True if the motor is currently moving

### kill()

Stop the motor

### move()

Initiate the motor to move to the setpoint

### property position

Control the angular position in counts, 4000 per revolution (int).

### property position\_error

Get the error in the number of counts that corresponds to the error in the angular position where 1 revolution equals 4000 counts

7.42. Parker 505

#### read()

Overwrite the Instrument.read command to provide the correct functionality

#### reset()

Reset the motor controller while blocking and (CAUTION) resets the absolute position value of the motor

### set\_defaults()

Set up the default values for the motor, which is run upon construction

#### set\_hardware\_limits(positive=True, negative=True)

Enable (True) or disables (False) the hardware limits for the motor

#### set\_software\_limits(positive, negative)

Set the software limits for motion based on the count unit where 4000 counts is 1 revolution

### property status

Get a list of the motor status in readable format

### stop()

Stop the motor during movement

### use\_absolute\_position()

Set the motor to accept setpoints from an absolute zero position

### use\_relative\_position()

Set the motor to accept setpoints that are relative to the last position

### property velocity

Set the velocity setpoint in revolutions per second.

### 7.43 Philips

This section contains specific documentation on the Philips instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.`.

### 7.43.1 Philips PM 6669 Universal Frequency Counter

class pymeasure.instruments.philips.PM6669(adapter, name='Philips PM 6669', \*\*kwargs)

Bases: Instrument

Represents the Philips PM 6669 instrument.

### property SRQMask

Control the SRQ mask

### property freerun\_enabled

Control the freerun settings

### property gate\_enabled

Control the gate

In the totalize function, set this to True to open the gate and to False to close the gate. The count can then be read with read\_measurement(). For most purposes you want to be in freerun mode.

#### property id

Get the instrument identification

### property measurement\_settings

Get the measurement settings from the device

#### property measurement\_time

Control the measurement time in seconds

#### property measurement\_timeout

Control the measurement timeout

this timeout only has meaning when freerun is off.

### property measuring\_function

Control the measuring function on the device (str).

#### read()

Read function with instrument bug work around.

If a request is made to the device while a measurement is ready, both the reply and the measurement are returned. The measurement is filtered out and put in a backlog Queue.

### read\_measurement(wait\_for\_srq=False)

Wait for an SRQ from the device and then reads the result.

If wait\_for\_srq is set, MSR to be set to MSRFlag.MEASUREMENT\_READY

### reset\_to\_defaults()

Reset the instruments to default settings

### spoll()

Read the status of the device.

### trigger()

Trigger the device when not in freerun mode.

### 7.44 Pendulum

This section contains specific documentation on the Pendulum instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.44.1 Pendulum CNT91 frequency counter

```
class pymeasure.instruments.pendulum.cnt91.CNT91(adapter, name='Pendulum CNT-91', **kwargs)
```

Bases: SCPIUnknownMixin, Instrument

Represents a Pendulum CNT-91 frequency counter.

### property batch\_size

Get maximum number of buffer entries that can be transmitted at once.

Record a time series to the buffer and read it out after completion.

7.44. Pendulum 507

#### **Parameters**

- channel Channel that should be used
- n\_samples The number of samples
- gate\_time Gate time in s
- trigger\_source Optionally specify a trigger source to start the measurement
- back\_to\_back If True, the buffer measurement is performed back-to-back.
- sample\_rate Sample rate in Hz

Deprecated since version 0.14: Use parameter *gate\_time* instead.

### configure\_frequency\_array\_measurement(n\_samples, channel, back\_to\_back=True)

Configure the counter for an array of measurements.

### **Parameters**

- n\_samples The number of samples
- channel Measurement channel (A, B, C, E, INTREF)
- back\_to\_back If True, the buffer measurement is performed back-to-back.

### property continuous

Control whether to perform continuous measurements.

#### property external\_arming\_start\_slope

Control slope for the start arming condition (str 'POS' or 'NEG').

### property external\_start\_arming\_source

Control external arming source ('A', 'B', 'E' (rear) or 'IMM' for immediately arming).

### property format

Control response format ('ASCII' or 'REAL').

#### property gate\_time

Control gate time of one measurement in s (float strictly from 2e-8 to 1000).

### property interpolator\_autocalibrated

Control if interpolators should be calibrated automatically (bool).

### property measurement\_time

Control gate time of one measurement in s (float strictly from 2e-8 to 1000).

Deprecated since version 0.14: Use gate time instead.

### read\_buffer(n=10000)

Read out n samples from the buffer.

#### **Parameters**

 ${\bf n}$  – Number of samples that should be read from the buffer. The maximum number of 10000 samples is read out by default.

#### Returns

Frequency values from the buffer.

### 7.45 Proterial

This section contains specific documentation on the Proterial (formerly Hitachi Metals) instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.45.1 ROD-4 MFC Controller

class pymeasure.instruments.proterial.ROD4(adapter, name='ROD-4 MFC Controller', \*\*kwargs)

```
Bases: Instrument
```

Represents the Proterial ROD-4(A) operator for mass flow controllers and provides a high-level interface for interacting with the instrument. User must specify which channel to control (1-4).

```
rod4 = ROD4("ASRL1::INSTR")

print(rod4.version)  # Print version and series number
rod4.ch_1.mfc_range = 500  # Sets Channel 1 MFC range to 500 sccm
rod4.ch_2.valve_mode = 'flow'  # Sets Channel 2 MFC to flow control
rod4.ch_3.setpoint = 50  # Sets Channel 3 MFC to flow at 50% of full range
print(rod4.ch_4.actual_flow)  # Prints Channel 4 actual MFC flow in %
```

ch\_1

#### Channel

ROD4Channel

ch\_2

#### Channel

ROD4Channel

 $ch_3$ 

### Channel

ROD4Channel

 $ch_4$ 

#### Channel

ROD4Channel

#### check\_errors()

Read all errors from the instrument and log them.

### Returns

List of error entries.

### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

7.45. Proterial 509

### check\_set\_errors()

Read 'OK' from ROD-4 after setting.

#### clear()

Clears the instrument status byte

### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

### property id

Get the identification of the instrument.

### property keyboard\_locked

Set the front keyboard lock status.

### property next\_error

Get the next error of the instrument (tuple of code and message).

### property options

Get the device options installed.

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

Read up to (excluding) *read\_termination* or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

### **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Resets the instrument.

### shutdown()

Brings the instrument to a safe and stable state

### property status

Get the status byte and Master Summary Status bit.

### property version

Get the version and series number. Returns x.xx<TAB>S/N

### wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

#### write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

### class pymeasure.instruments.proterial.rod4.ROD4Channel(parent, id)

Bases: Channel

Implementation of a ROD-4 MFC channel.

### property actual\_flow

Measure the actual flow in %.

### property flow\_unit\_display

Set the flow units on the front display. Valid options are %, sccm, or slm. Display in absolute units is in sccm for control range < 10 slm.

### property mfc\_range

Control the MFC range in sccm. Upper limit is 200 slm.

#### property ramp\_time

Control the MFC setpoint ramping time in seconds.

### property setpoint

Control the setpoint in % of MFC range.

### property valve\_mode

Control the MFC valve mode. Valid options are flow, close, and open.

## 7.46 PTW Freiburg

This section contains specific documentation on the PTW Freiburg GmbH instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

7.46. PTW Freiburg 511

### 7.46.1 PTW DIAMENTOR DAP dosemeters

Bases: Instrument

A class representing the PTW DIAMENTOR DAP dosemeters.

### property baudrate

Control the baudrate.

Type

int, strictly in 9600, 19200, 38400, 57600, 115200

The baudrate is changed after sending the respone.

### property calibration\_factor

Control the calibration factor of the measurement chamber in  $\mu Gy^*m^2/s$ .

**Type** 

float, strictly from 1E8 to 9.999E12, default: 1.0E9

The unit of the calibration factor is always  $\mu Gy^*m^2/s$ . It is independent from the selected dap\_unit.

### **A** Warning

Changing the calibration factor can lead to wrong measurements!

The calibration factor of the last calibration can be found on a label on the case or in the calibration certificate of the DIAMENTOR chamber.

### check\_set\_errors()

Check for errors after sending a command.

### property constancy\_check\_passed

Get the DIAMENTOR electrical constancy check result (bool).

### property correction\_factor

Control the correction factor of the chamber.

**Type** 

float, strictly from 0 to 9.999, default: 1.0

### property dap\_unit

Control the dose-area-product (DAP) unit.

**Type** 

str, strictly in cGycm2, Gycm2, uGym2, Rcm2

- cGycm2 selects cGy\*cm<sup>2</sup>
- Gycm2 selects Gy\*cm<sup>2</sup>
- uGym2 selects µGy\*m<sup>2</sup>
- Rcm2 selects R\*cm<sup>2</sup>

### execute\_selftest()

Execute the DIAMENTOR selftest.

#### Raises

ValueError if selftest fails

### property id

Get the firmware version (str) ("CRS x.xx")

### property is\_calibrated

Get the calibration status (bool).

#### property is\_eeprom\_ok

Get the EEPROM CRC ok status (bool).

#### property measurement

Get the measurement result.

#### Returns

dict

### Dict keys

dap, dap\_rate, time

The result consists of the dose-area-product (DAP) and the dose-area-product rate. The units of dap and dap\_rate depend on the *dap\_unit* property. Time is in seconds.

### property pressure

Control the atmospheric pressure in hPa.

#### Type

int, strictly from 500 to 1500, default: 1013

It is used for the air density correction.

#### read()

Read the device response and check for errors.

### Returns

str

### Raises

ValueError for error response or ConnectionError for an unknown error

### reset()

Reset the dose and charge measurement values.

### property serial\_number

Get the serial number (int).

### property temperature

Control the chamber temperature in degree Celsius.

#### Type

int, strictly from 0 to 70, default: 20

It is used for the air density correction.

7.46. PTW Freiburg 513

### 7.46.2 PTW UNIDOS Tango/Romeo dosemeters

class pymeasure.instruments.ptw.ptwUNIDOS(adapter, name='PTW UNIDOS dosemeter', \*\*kwargs)

Bases: Instrument

A class representing the PTW UNIDOS Tango/Romeo dosemeters.

### property autoreset\_enabled

Control the measurement autoreset function (bool).

### property autostart\_enabled

Control the measurement autostart function (bool).

### property autostart\_level

Control the threshold level of autostart measurements (str strictly in "LOW", "MEDIUM", "HIGH").

str, strictly in LOW, MEDIUM, HIGH

### check\_set\_errors()

Check for errors after sending a command.

clear()

Clear the complete measurement history.



Write permission is required.

### property electrical\_units\_enabled

Control whether electrical units are used (bool).

### errorflags\_to\_text(flags)

Convert the error flags to the error message(s).

### **Parameters**

**flags** (str)

hold()

Set the measurment to HOLD state.



1 Note

Write permission is required.

### property id

Get the dosemeter ID.

Returns

list of str

### property integration\_time

Control the time of the interval measurement in seconds (strictly from 1 to 3599999).

**Type** 

int, strictly from 1 to 3599999

### intervall(intervall=None)

Execute an intervall measurement.

### **Parameters**

**intervall** (*int*) – optional, measurement intervall in seconds

If *intervall* is not specified a measurement with the current setting of the *integration\_time* is executed.



Write permission is required.

### property lan\_config

Get the ethernet configuration.

Returns

dict

Dict kevs

dns, ipv4, ipv6

### property mac\_address

Get the dosemeter MAC address.

#### Returns

str

### measure()

Start the dose or charge measurement.



### **1** Note

Write permission is required.

### property measurement\_history

Get the measurement history.

### **Returns**

list of dict

### Dict kevs

date, detector, dose, flags, id, kTotal

### property measurement\_parameters

Get the measurement parameters.

### Returns

dict

### Dict keys

```
highResolution,
                                                       integrationTime,
correction,
              detectorGuid,
                                                 hv,
                                                         limitDoseRate,
isElectrical.
                  limitDose,
                                  limitDoseEnabled,
limitDoseRateEnabled,
                           range,
                                        triggerAuto,
                                                          triggerReset,
triggerSensitivity
```

7.46. PTW Freiburg 515

### property measurement\_result

Get the measurement results.

### Returns

dict

#### Dict keys

status, charge, dose, current, doserate, timebase, time, voltage, error

### property range

Control the measurement range (str strictly in "VERY\_LOW", "LOW", "MEDIUM", "HIGH").

#### Type

str, strictly in VERY\_LOW, LOW, MEDIUM, HIGH

### property range\_max

Get the max value of the current measurement range.

#### Returns

dict

#### Dict keys

range, current, doserate, timebase

### property range\_res

Get the resolution of the measurement range.

#### Returns

dict

### Dict keys

range, charge, dose, current, doserate, timebase

### read()

Read the device response and check for errors.

#### Returns

Read string with semicolons replaced by comma

#### Raises

ValueError for error response or ConnectionError for an unknown error

### read\_detector(guid='ALL')

Read the properties of the requested detector.

#### **Parameters**

**guid** (*str*) – optional, ID of the detector. A list of all detectors is returned if *guid* is not passed.

### Type

dict or list of dict

### Dict keys

calibrationFactor, calibrationFactorUnit, calibrationLab, calibrationProcedure, calibrationTemp, calibrationType, calibrationUser, comment, guid, manufacturer, maxHv, measuredQuantity, minHv, name, nominalHv, polarity, radiationQuality, range, serialNumber, testDate, testLimits, testResult, testTarget, timebase, type, typeNumber

### reset()

Reset the dose and charge measurement values.



### 1 Note

Write permission is required.

#### selftest()

Execute the electrometer selftest.

The function returns before the end of the selftest. End and result of the self test have to be requested by the selftest\_result property.



#### 1 Note

Write permission is required.

### property selftest\_result

Get the dosemeter selftest status and result.

### Returns

dict

### Dict keys

status, time\_remaining, time\_total, low, medium, high

### property serial\_number

Get the dosemeter serial number.

#### Returns

int

### property status

Get the measurement status.

### Returns

str

The status string has of one of the following values: RES, MEAS, HOLD, INT, INTHLD, ZERO, AUTO, AUTO\_MEAS, AUTO\_HOLD, EOM, WAIT, INIT, ERROR, SELF\_TEST or TST

#### property system\_info

Get the system information.

#### Returns

dict

### Dict keys

calibrationDate, debugBuild, features, hardwareVersion, hostname, lastTest, macAddress, nextCalibration, serialNumber, softwareVersion, testTemperature, typeNumber

### property system\_settings

Get the system settings.

#### Returns

dict

7.46. PTW Freiburg 517

### Dict keys

brightness, date, keyboardSound, locale, mark, startAsMaster, time, timezone, volume

### property tfi

Get the telegram failure information.

### Returns

str

The property provides information about the last failed command with HTTP request.

### property voltage

Control the detector voltage in Volts.

#### Type

int, strictly from -400 to 400 and within detector limits

The Limits of the detector are applied.

### property wlan\_config

Get the WLAN access point configuration.

#### Returns

dict

### Dict keys

enabled, ssid

### property write\_enabled

Control the write permission (bool).

The result of the request/release has to be checked afterwards.

### Important

The UNIDOS has to be in viewer mode (closed lock symbol on the display).

### zero()

Execute a zero correction measurement.

The function returns before the end of the zero correction measurement. The end of the zero correction measurement has to be requested by the zero\_status property.



### 1 Note

Write permission is required.

### property zero\_status

Get the status of the zero correction measurement.

### Returns

dict

### Dict keys

status, time\_remaining, time\_total

### 7.47 Racal-Dana

This section contains specific documentation on the Raca-Dana instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.47.1 Racal-Dana 1992 Universal Counter

class pymeasure.instruments.racal.Racal1992(adapter, name='Racal-Dana 1992', \*\*kwargs)

Bases: Instrument

Represents the Racal-Dana 1992 Universal counter

```
from pymeasure.instruments.racal import Racal1992
counter = Racal1992("GPIB0::10")
```

This class should also work for Racal-Dana 1991, it has the same product manual, as long as you don't use functionality that requires channel B.

channel\_settings(channel\_name, \*\*settings)

Set channel configuration paramters.

#### **Parameters**

- channel\_name 'A' or 'B'
- **settings** one or multiple of the following:

'coupling': 'AC' or 'DC' 'attenuation': 'X1' or 'X10' 'trigger': 'auto' or 'manual' 'impedance': '50' or '1M' 'slope': 'pos' or 'neg' 'filtering': True or False (only allowed for channel A) 'input\_select': 'separate' or 'common' (only allowed for channel B) 'trigger\_level': <floating point number>

**static decode**(*v*, *allowed types=None*)

Decode received message.

All values returned follow the same format: 2 letters to indicate the type of the value returned, followed by a floating point number (which could be an integer, of course.) This here, for example, is math constant Z: MZ+001.0000000E+00

### property delay\_enable

Control delay. True=enable, False=disable

#### property delay\_time

Control delay time.

### property device\_type

Get unit device type. Should return 1992 for a Racal-Dana 1992 or 1991 for a Racal-Dana 1991.

### property gpib\_software\_version

Get GPIB software version

### property math\_mode

Set math mode. True=enable, False=disable

### property math\_x

Control math constant X.

7.47. Racal-Dana 519

#### property math\_z

Control math constant Z.

### property measured\_value

Get measured value.

A Racal-Dana 1992 doesn't return measurement data after a request for measurement data. Instead, it fills a FIFO with data whenever it completes a measurement. When the FIFO is full, the oldest measurement is removed.

The FIFO buffer gets cleared when a command is received that requires an immediate reply, such reading a setting. It also gets cleared when an operating mode is cleared.

When there is no measurement data, this property will stall until data is available. It will also timeout after a time that can be set with the standard pyvisa API.

One can make sure that measurement data is available by first calling wait\_for\_measurement().

### property operating\_mode

Set operating mode.

#### Permitted modes are:

```
'self_check', 'frequency_a', 'period_a', 'phase_a_rel_b', 'ratio_a_to_b', 'ratio_c_to_b', 'interval_a_to_b', 'total_a_by_b', 'frequency_c'
```

### preset()

Configure instrument with default presets.

#### read()

Read up to (excluding) *read\_termination* or the whole read buffer.

### reset\_measurement()

Reset ongoing measurement.

#### property resolution

Control the resolution of the counter with an integer from 3 to 10 that specifies the number of significant digits.

#### property software\_version

Get instrument software version

### property special\_function\_enable

Control special function. True=enable, False=disable

#### property special\_function\_number

Control special function.

### property total\_so\_far

Get total number of events so far.

### property trigger\_level\_a

Control trigger level for channel A

#### property trigger\_level\_b

Control trigger level for channel B

### wait\_for\_measurement(timeout=None, progressDots=False)

Wait until a new measurement is available.

#### **Parameters**

- **timeout** number of seconds to wait before timeout exception.
- progressDots when true, print '.' after each ready-check

### write(s)

Add a space in front of all commands that are sent to the instrument to work around weird model issue.

It shouldn't be needed on almost all devices, but it also doesn't hurt. And it fixes a real issue that's seen on a few devices.

### 7.48 Razorbill

This section contains specific documentation on the Razorbill instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.48.1 Razorbill RP100 custrom power supply for Razorbill Instrums stress & strain cells

Bases: SCPIUnknownMixin. Instrument

Represents Razorbill RP100 strain cell controller

```
scontrol = razorbillRP100("ASRL/dev/ttyACM0::INSTR")
scontrol.output_1 = True  # turns output on
scontrol.slew_rate_1 = 1  # sets slew rate to 1V/s
scontrol.voltage_1 = 10  # sets voltage on output 1 to 10V
```

### property contact\_current\_1

Get the current in amps present at the front panel output of channel 1

### property contact\_current\_2

Get the current in amps present at the front panel output of channel 2

### property contact\_voltage\_1

Get the Voltage in volts present at the front panel output of channel 1

#### property contact\_voltage\_2

Get the Voltage in volts present at the front panel output of channel 2

### property instant\_voltage\_1

Get the instantaneous output of source one in volts

### property instant\_voltage\_2

Get the instantaneous output of source two in volts

#### property output\_1

Control output of channel 1 on or off

### property output\_2

Control output of channel 2 on or off

7.48. Razorbill 521

#### property slew\_rate\_1

Control the source slew rate in volts/sec of channel 1

### property slew\_rate\_2

Control the source slew rate in volts/sec of channel 2

### property voltage\_1

Control the output voltage of channel 1

### property voltage\_2

Control the output voltage of channel 2

### 7.49 Redpitaya

This section contains specific documentation on the Redpitaya instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.49.1 Redpitaya board for analog signal acquisition and generation as well as digital input/output

```
class pymeasure.instruments.redpitaya.redpitaya_scpi.RedPitayaScpi(adapter=None, ip\_address: str = '169.254.134.87', port: int = 5000, name='Redpitaya SCPI', read_termination=\r\n', write_termination=\r\n', write_termination=\r\n', **kwargs)
```

Bases: SCPIMixin, Instrument

This is the class for the Redpitaya reconfigurable board

The instrument is accessed using a TCP/IP Socket communication, that is an adapter in the form: "TCPIP::x.y.z.k::port::SOCKET" where x.y.z.k is the IP address of the SCPI server (that should be activated on the board) and port is the TCP/IP port number, usually 5000

To activate the SCPI server, you have to connect first the redpitaya to your computer/network and enter the url address written on the network plug (on the redpitaya). It should be something like "RP-F06432.LOCAL/" then browse the menu, open the Development application and activate the SCPI server. When activating the server, you'll be notified with the IP/port address to use with this Instrument.

#### **Parameters**

- ip\_address IP address to use, if *adapter* is None.
- **port** Port number to use, if *adapter* is None.

### analog\_in

#### Channels

ain1: AnalogInputFastChannel, ain2: AnalogInputFastChannel

### analog\_in\_slow

#### Channels

```
ainslow0: AnalogInputSlowChannel, ainslow1: AnalogInputSlowChannel,
ainslow2: AnalogInputSlowChannel, ainslow3: AnalogInputSlowChannel
```

#### analog\_out\_slow

#### Channels

aoutslow0: AnalogOutputSlowChannel, aoutslow1: AnalogOutputSlowChannel, aoutslow2: AnalogOutputSlowChannel, aoutslow3: AnalogOutputSlowChannel

#### dioN

#### Channels

dioN0: DigitalChannelN, dioN1: DigitalChannelN, dioN2: DigitalChannelN, dioN3: DigitalChannelN, dioN4: DigitalChannelN, dioN5: DigitalChannelN, dioN6: DigitalChannelN

#### dioP

#### Channels

dioP0: DigitalChannelP, dioP1: DigitalChannelP, dioP2: DigitalChannelP, dioP3: DigitalChannelP, dioP4: DigitalChannelP, dioP5: DigitalChannelP, dioP6: DigitalChannelP

#### led

#### Channels

led0: DigitalChannelLed, led1: DigitalChannelLed, led2: DigitalChannelLed, led3: DigitalChannelLed, led4: DigitalChannelLed, led5: DigitalChannelLed, led6: DigitalChannelLed, led7: DigitalChannelLed

### property acq\_buffer\_filled

Get the status of the buffer(bool), if True the buffer is full

### property acq\_format

Set the format of the retrieved buffer data (str), either 'BIN', or 'ASCII' (default)

### property acq\_trigger\_delay\_ns

Control the trigger delay in nanoseconds (int) in the range [-8192, 8192] / CLOCK

#### property acq\_trigger\_delay\_samples

Control the trigger delay in number of samples (int) in the range [-8192, 8192]

### property acq\_trigger\_level

Control the level of the trigger in volts The allowed range should be dynamically set depending on the gain settings either +-LV\_MAX or +- HV\_MAX (dynamic)

### property acq\_trigger\_position

Get the position within the buffer where the trigger event happened

#### property acq\_trigger\_source

Set the trigger source (str), one of RedPitayaScpi.TRIGGER\_SOURCES. PE and NE means respectively Positive and Negative edge

### property acq\_trigger\_status

Get the trigger status (bool), if True the trigger as been fired (or is disabled)

#### property acq\_units

Control the output data units (str), either 'RAW', or 'VOLTS' (default)

### analog\_reset()

Reset the voltage of all analog channels

7.49. Redpitaya 523

### property average\_skipped\_samples

Control the use of skipped samples (if decimation > 1) to average the returned acquisition array (bool)

### property board\_name

Get the RedPitaya board name

### property buffer\_length

Measure the size of the buffer, that is the number of points of the acquisition

### property date

Control the date on board date should be given as a datetime.date object

### property decimation

Control the decimation (int) as  $2^{**}n$  with n in range [0, 16] The sampling rate is given as 125MS/s / decimation

### digital\_reset()

Reset the state of all digital lines

### property time

Control the time on board time should be given as a datetime.time object

### **7.50 Rigol**

This section contains specific documentation on the Rigol instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.50.1 Rigol DG800 waveform generator

```
class pymeasure.instruments.rigol.DG800(adapter, name='DG800', **kwargs)
```

Bases: SCPIMixin, Instrument

Represents a Rigol DG800-series waveform generator and provides a high-level for interacting with the instrument

### channel\_1

### Channel

VoltageChannel

### channel\_2

### Channel

VoltageChannel

class pymeasure.instruments.rigol.rigol\_dg800.VoltageChannel(parent, id)

Bases: Channel

Represents a channel of the signal generator.

### property dc

Set the waveform generator to output a DC voltage in V.

#### property duty\_cycle

Control the duty cycle of pulses out of the waveform generator. (float)

### property frequency

Control the output frequency (Hz)

### property high\_impedance

Control the output impedance to be HighZ. This means displayed voltages will be internally converted to correspond that seen by an infinite load, not 50 Ohms. See p. 82 of the manual for more information.

### property load

Control the output impedance of a given channel. Returns 9.9e+37 for infinite impedance.

### property noise

Set the waveform generator to output noise of specified parameters :param ampl: (float) The peak to peak amplitude of the noise in V :param offset: (float) The DC offset of the noise in V

### property output\_enabled

Control the status of the channel. True if the channel is on and False if not.

### property pulse

Set the waveform generator to output a pulse wave of specified parameters Note that the duty cycle is specified using the duty\_cycle method. :param int freq: The frequency of the wave in Hz :param float ampl: The peak to peak amplitude of the wave in V :param float offset: The DC offset of the wave in V :param float phase: The phase offset of the wave in degrees

### property pulse\_width

Control the pulse width of pulses (in seconds) out of the waveform generator.

### property shape

Control the waveform type of the specified channel.

### property sine

Set the waveform generator to output a sine of specified parameters :param int freq: The frequency of the sine in Hz :param float ampl: The peak to peak amplitude of the sine in V :param float offset: The DC offset of the sine in V :param float phase: The phase offset of the sine in degrees

#### property square

Set the waveform generator to output a square wave of specified parameters :param int freq: The frequency of the wave in Hz :param float ampl: The peak to peak amplitude of the wave in V :param float offset: The DC offset of the wave in V :param float phase: The phase offset of the wave in degrees

### property sync\_enabled

Control the synchronization flag. True if the channel triggers the synchronization connector and False if not

### property triangle

Set the waveform generator to output a triangle wave of specified parameters :param int freq: The frequency of the wave in Hz:param float ampl: The peak to peak amplitude of the wave in V:param float offset: The DC offset of the wave in V:param float phase: The phase offset of the wave in degrees

### property waveform

Get a descriptor of the waveform applied.

7.50. Rigol 525

### 7.51 Rohde & Schwarz

This section contains specific documentation on the Rohde & Schwarz instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.51.1 R&S SFM TV test transmitter

class pymeasure.instruments.rohdeschwarz.sfm.SFM(adapter, name='Rohde&Schwarz SFM', \*\*kwargs)

Bases: SCPIMixin, Instrument

Represents the Rohde&Schwarz SFM TV test transmitter interface for interacting with the instrument.

### **1** Note

The current implementation only works with the first system in this unit.

Further source extension for system 2-6 would be required.

The intermodulation subsystem is also not yet implemented.

### property R75\_out

A bool property that controls the use of the 75R output (if installed)

Value	Meaning
False	50R output active (N)
True	75R output active (BNC)

refer also to chapter 3.6.5 of the manual

### property TV\_country

A string property that controls the country specifics of the video/sound system to be used Possible values are:

Value	Meaning
BG_G	BG General
DK_G	DK General
I_G	I General
L_G	L General
GERM	Germany
BELG	Belgium
NETH	Netherlands
FIN	Finland
AUST	Australia
BG_T	BG Th
DENM	Denmark
NORW	Norway
SWED	Sweden
GUS	Russia
POL1	Poland
POL2	Poland
HUNG	Hungary
CHEC	Czech Republic
CHINA1	China
CHINA2	China
GRE	Great Britain
SAFR	South Africa
FRAN	France
USA	United States
KOR	Korea
JAP	Japan
CAN	Canada
SAM	South America

Please confirm with the manual about the details for these settings.

### property TV\_standard

A string property that controls the type of video standard

Possible values are:

Value	Lines	System
BG	625	PAL
DK	625	SECAM
I	625	PAL
K1	625	SECAM
L	625	SECAM
M	525	NTSC
N	625	NTSC

Please confirm with the manual about the details for these settings.

### property basic\_info

A String property containing information about the hardware modules installed in the unit

7.51. Rohde & Schwarz 527

### property beeper\_enabled

A bool property that controls the beeper status,

refer also to chapter 3.6.8 of the manual

### calibration(number=1, subsystem=None)

Function to either calibrate the whole modulator, when subsystem parameter is omitted, or calibrate a subsystem of the modulator.

Valid subsystem selections: "NICam, VISion, SOUNd1, SOUNd2, CODer"

### channel\_down\_relative()

Decreases the output frequency to the next low channel/special channel based on the current country settings

### property channel\_sweep\_start

A float property controlling the start frequency for channel sweep in Hz

- Minimum 5 MHz
- · Maximum 1 GHz

### property channel\_sweep\_step

A float property controlling the start frequency for channel sweep in Hz

- Minimum 5 MHz
- · Maximum 1 GHz

### property channel\_sweep\_stop

A float property controlling the start frequency for channel sweep in Hz

- Minimum 5 MHz
- Maximum 1 GHz

### property channel\_table

A string property controlling which channel table is used

Possible selections are:

Value	Meaning
DEF	Default channel table
USR1	User table No. 1
USR2	User table No. 2
USR3	User table No. 3
USR4	User table No. 4
USR5	User table No. 5

refer also to chapter 3.6.6.1 of the manual

#### channel\_up\_relative()

Increases the output frequency to the next higher channel/special channel based on the current country settings

### coder\_adjust()

Starts the automatic setting of the differential deviation

refer also to chapter 3.6.6.4 of the manual

### property coder\_id\_frequency

A int property that controls the frequency of the identification of the coder

valid range 0 .. 200 Hz

### property coder\_modulation\_degree

A float property that controls the modulation degree of the identification of the coder

valid range: 0 .. 0.9

### property coder\_pilot\_deviation

A int property that controls deviation of the pilot frequency of the coder

valid range: 1 .. 4 kHz

### property coder\_pilot\_frequency

A int property that controls the pilot frequency of the coder

valid range: 40 .. 60 kHz

### property cw\_frequency

A float property controlling the CW-frequency in Hz

- Minimum 5 MHz
- · Maximum 1 GHz

### property date

A list property for the date of the RTC in the unit

### property event\_reg

Content of the event register of the Status Operation Register refer also to chapter 3.6.7 of the manual

### property ext\_ref\_base\_unit

A bool property for the external reference for the basic unit

Value	Meaning
False	Internal 10 MHz is used
True	External 10 MHz is used

#### property ext\_ref\_extension

A bool property for the external reference for the extension frame

Value	Meaning
False	Internal 10 MHz is used
True	External 10 MHz is used

### property ext\_vid\_connector

A string property controlling which connector is used as the input of the video source

Possible selections are:

7.51. Rohde & Schwarz

Value	Meaning
HIGH	Front connector - Hi-Z
LOW	Front connector - 75R
REAR1	Rear connector 1
REAR2	Rear connector 2
AUTO	Automatic assignment

### property external\_modulation\_frequency

A int property that controls the setting for the external modulator frequency

valid range: 32 .. 46 MHz

### property external\_modulation\_power

A int property that controls the setting for the external modulator output power

valid range: -7..0 dBm

refer also to chapter 3.6.6.5 of the manual

### property external\_modulation\_source

A bool property for the modulation source selection

refer also to chapter 3.6.6.8 of the manual

### property frequency

A float property controlling the frequency in Hz

- Minimum 5 MHz
- · Maximum 1 GHz

### property frequency\_mode

A string property controlling which the unit is used in

Possible selections are:

Value	Meaning
CW	Continuous wave mode
FIXED	fixed frequency mode
CHSW	Channel sweep
RFSW	Frequency sweep



### 1 Note

selecting the sweep mode, will start the sweep imemdiately!

### property gpib\_address

A int property that controls the GPIB address of the unit

valid range: 0..30

### property high\_frequency\_resolution

A property that controls the frequency resolution,

Possible selections are:

Value	Meaning
False	Low resolution (1000Hz)
True	High resolution (1Hz)

### property level

A float property controlling the output level in dBm,

- Minimum -99dBm
- Maximum 10dBm (depending on output mode)

refer also to chapter 3.6.6.2 of the manual

#### property level\_mode

A string property controlling the output attenuator and linearity mode

Possible selections are:

Value	Meaning	max. output level
NORM	Normal mode	+6 dBm
LOWN	low noise mode	+10 dBm
CONT	continuous mode	+10 dBm
LOWD	low distortion mode	+0 dBm

Continuous mode allows up to 14 dB of level setting without use of the mechanical attenuator.

### property lower\_sideband\_enabled

A bool property that controls the use of the lower sideband

refer also to chapter 3.6.6.10 of the manual

### property modulation\_enabled

A bool property that controls the modulation status

### property nicam\_IQ\_inverted

A bool property that controls if the NICAM IQ signals are inverted or not

Value	Meaning
False	normal (IQ)
True	inverted (QI)

### property nicam\_additional\_bits

A int property that controls the additional data in the NICAM modulator

valid range: 0 .. 2047

7.51. Rohde & Schwarz 531

### property nicam\_audio\_frequency

A int property that controls the frequency of the internal sound generator

valid range: 0 Hz .. 15 kHz

### property nicam\_audio\_volume

A float property that controls the audio volume in the NICAM modulator in dB

valid range: 0..60 dB

### property nicam\_bit\_error\_enabled

A bool property that controls the status of an artificial bit error rate to be applied

### property nicam\_bit\_error\_rate

A float property that controls the artificial bit error rate.

valid range: 1.2E-7 .. 2E-3

### property nicam\_carrier\_enabled

A bool property that controls if the NICAM carrier is switched on or off

### property nicam\_carrier\_frequency

A float property that controls the frequency of the NICAM carrier

valid range: 33.05 MHz +/- 0.2 Mhz

### property nicam\_carrier\_level

A float property that controls the value of the NICAM carrier

valid range: -40 .. -13 dB

### property nicam\_control\_bits

A int property that controls the additional data in the NICAM modulator

valid range: 0 .. 3

### property nicam\_data

A int property that controls the data in the NICAM modulator

valid range: 0 .. 2047

### property nicam\_intercarrier\_frequency

A float property that controls the inter-carrier frequency of the NICAM carrier

valid range: 5 .. 9 MHz

#### property nicam\_mode

A string property that controls the signal type to be sent via NICAM

Possible values are:

Value	Meaning
MON	Mono sound + NICAM data
STER	Stereo sound
DUAL	Dual channel sound
DATA	NICAM data only

refer also to chapter 3.6.6.6 of the manual

### property nicam\_preemphasis\_enabled

A bool property that controls the status of the J17 preemphasis

### property nicam\_source

A string property that controls the signal source for NICAM

Possible values are:

Value	Meaning
INT	Internal audio generator(s)
EXT	External audio source
CW	Continuous wave signal
RAND	Random data stream
TEST	Test signal

### property nicam\_test\_signal

A int property that controls the selection of the test signal applied

Value	Meaning
1	Test signal 1 (91 kHz square wave, I&Q 90deg apart)
2	Test signal 2 (45.5 kHz square wave, I&Q 90deg apart)
3	Test signal 3 (182 kHz sine wave, I&Q in phase)

### property normal\_channel

A int property controlling the current selected regular/normal channel number valid selections are based on the country settings.

### property operation\_enable\_reg

Content of the enable register of the Status Operation Register

Valid range: 0...32767

### property output\_voltage

A float property controlling the output level in Volt,

Minimum 2.50891e-6, Maximum 0.707068 (depending on output mode) refer also to chapter 3.6.6.12 of the manual

### property questionable\_event\_reg

Content of the event register of the Status Questionable Operation Register

### property questionable\_operation\_enable\_reg

Content of the enable register of the Status Questionable Operation Register

Valid range 0...32767

### property questionanble\_status\_reg

Content of the condition register of the Status Questionable Operation Register

### property remote\_interfaces

A string property controlling the selection of interfaces for remote control

Possible selections are:

7.51. Rohde & Schwarz 533

Value	Meaning
OFF	no remote control
GPIB	GPIB only enabled
SER	RS232 only enabled
BOTH	GPIB & RS232 enabled

### property rf\_out\_enabled

A bool property that controls the status of the RF-output

### property rf\_sweep\_center

A float property controlling the center frequency for sweep in Hz

- Minimum 5 MHz
- Maximum 1 GHz

### property rf\_sweep\_span

A float property controlling the sweep span in Hz,

- Minimum 1 kHz
- Maximum 1 GHz

### property rf\_sweep\_start

A float property controlling the start frequency for sweep in Hz

- Minimum 5 MHz
- Maximum 1 GHz

### property rf\_sweep\_step

A float property controlling the stepwidth for sweep in Hz,

- Minimum 1 kHz
- Maximum 1 GHz

### property rf\_sweep\_stop

A float property controlling the stop frequency for sweep in Hz

- Minimum 5 MHz
- · Maximum 1 GHz

### property scale\_volt

A string property that controls the unit to be used for voltage entries on the unit

Possible values are: AV,FV, PV, NV, UV, MV, V, KV, MAV, GV, TV, PEV, EV, DBAV, DBFV, DBPV, DBNV, DBW, DBV, DBKV, DBMAV, DBGV, DBTV, DBPEV, DBEV

refer also to chapter 3.6.9 of the manual

### property serial\_baud

A int property that controls the serial communication speed,

Possible values are: 110,300,600,1200,4800,9600,19200

#### property serial\_bits

A int property that controls the number of bits used in serial communication

Possible values are: 7 or 8

### property serial\_flowcontrol

A string property that controls the serial handshake type used in serial communication

Possible values are:

Value	Meaning
NONE	no flow-control/handshake
XON	XON/XOFF flow-control
ACK	hardware handshake with RTS&CTS

#### property serial\_parity

A string property that controls the parity type used for serial communication

Possible values are:

Value	Meaning
NONE	no parity
<b>EVEN</b>	even parity
ODD	odd parity
ONE	parity bit fixed to 1
ZERO	parity bit fixed to 0

#### property serial\_stopbits

A int property that controls the number of stop-bits used in serial communication,

Possible values are: 1 or 2

### property sound\_mode

A string property that controls the type of audio signal

Possible values are:

Value	Meaning
MONO	MOnoaural sound
PIL	pilot-carrier + mono
BTSC	BTSC + mono
STER	Stereo sound
DUAL	Dual channel sound
NIC	NICAM + Mono

### property special\_channel

A int property controlling the current selected special channel number valid selections are based on the country settings.

## property status\_info\_shown

A bool property that controls if the display shows information during remote control

### status\_preset()

partly resets the SCPI status reporting structures

### property status\_reg

Content of the condition register of the Status Operation Register

7.51. Rohde & Schwarz 535

#### property subsystem\_info

A String property containing information about the system configuration

## property system\_number

A int property for the selected systems (if more than 1 available)

- Minimum 1
- Maximum 6

## property time

A list property for the time of the RTC in the unit

### property vision\_average\_enabled

A bool property that controls the average mode for the vision system

### property vision\_balance

A float property that controls the balance of the vision modulator

valid range: -0.5 .. 0.5

## property vision\_carrier\_enabled

A bool property that controls the vision carrier status

refer also to chapter 3.6.6.9 of the manual

### property vision\_carrier\_frequency

A float property that controls the frequency of the vision carrier

valid range: 32 .. 46 MHz

#### property vision\_clamping\_average

A float property that controls the operation point of the vision modulator

valid range: -0.5 .. 0.5

### property vision\_clamping\_enabled

A bool property that controls the clamping behavior of the vision modulator

### property vision\_clamping\_mode

A string property that controls the clamping mode of the vision modulator

Possible selections are HARD or SOFT

#### property vision\_precorrection\_enabled

A bool property that controls the precorrection behavior of the vision modulator

## property vision\_residual\_carrier\_level

A float property that controls the value of the residual carrier

valid range: 0 .. 0.3 (30%)

### property vision\_sideband\_filter\_enabled

A bool property that controls the use of the VSBF (vestigal sideband filter) in the vision modulator

### property vision\_videosignal\_enabled

A bool property that controls if the video signal is switched on or off

### class pymeasure.instruments.rohdeschwarz.sfm.Sound\_Channel(instrument, number)

Bases: object

Class object for the two sound channels

refer also to chapter 3.6.6.7 of the user manual

### property carrier\_enabled

A bool property that controls if the audio carrier is switched on or off

## property carrier\_frequency

A float property that controls the frequency of the sound carrier

valid range: 32 .. 46 MHz

## property carrier\_level

A float property that controls the level of the audio carrier in dB relative to the vision carrier (0dB)

valid range: -34 .. -6 dB

#### property deviation

A int property that controls deviation of the selected audio signal

valid range: 0 .. 110 kHz

### property frequency

A int property that controls the frequency of the internal sound generator

valid range: 300 Hz .. 15 kHz

### property modulation\_degree

A float property that controls the modulation depth for the audio signal (Note: only for the use of AM in Standard L)

valid range: 0 .. 1 (100%)

### property modulation\_enabled

A bool property that controls the audio modulation status

Value	Meaning	
False	modulation disabled	
True	modulation enabled	

### property preemphasis\_enabled

A bool property that controls if the preemphasis for the audio is switched on or off

## property preemphasis\_time

A int property that controls if the mode of the preemphasis for the audio signal

Value	Meaning	
50	50 us preemphasis	
75	75 us preemphasis	

7.51. Rohde & Schwarz

#### property use\_external\_source

A bool property for the audio source selection

Value	Meaning
False	Internal audio generator(s)
True	External signal source

values(command, \*\*kwargs)

Reads a set of values from the instrument through the adapter, passing on any keyword arguments.

## 7.51.2 R&S FS Series spectrum analyzer

### Connecting to an R&S FSL via network

Once connected to the network, the instrument's IP address can be found by clicking the "Setup" button and navigating to "General Settings" -> "Network Address".

It can then be connected like this:

```
from pymeasure.instruments.rohdeschwarz import FSL
fsl = FSL("TCPIP::192.168.1.123::INSTR")
```

### Getting and setting parameters

Most parameters are implemented as properties, which means they can be read and written (getting and setting) in a consistent and simple way. If numerical values are provided, base units are used (s, Hz, dB, ...). Alternatively, the values can also be provided with a unit, e.g. "1.5 GHz" or "1.5GHz". Return values are always numerical.

```
# Getting the current center frequency
fsl.freq_center
9000000000.0
```

```
# Changing it to 10 MHz by providing the numerical value
fsl.freq_center = 10e6
```

```
# Verifying:
fsl.freq_center
10000000.0
```

```
# Changing it to 9 GHz by providing a string and verifying the result
fsl.freq_center = '9GHz'
fsl.freq_center
9000000000.0
```

```
# Setting the span to maximum
fsl.freq_span = '7 GHz'
```

### Reading a trace

We will read the current trace

```
x, y = fsl.read_trace()
```

#### **Markers**

Markers are implemented as their own class. You can create them like this:

```
m1 = fsl.create_marker()
```

Set peak exursion:

```
m1.peak_excursion = 3
```

Set marker to a specific position:

```
\boxed{\mathbf{m1.x} = 10e9}
```

Find the next peak to the left and get the level:

```
m1.to_next_peak('left')
m1.y
-34.9349060059
```

### **Delta markers**

Delta markers can be created by setting the appropriate keyword.

```
d2 = fsl.create_marker(is_delta_marker=True)
d2.name
'DELT2'
```

### **Example program**

Here is an example of a simple script for recording the peak of a signal.

```
m1 = fsl.create_marker() # create marker 1

# Set standard settings, set to full span
fsl.continuous_sweep = False
fsl.freq_span = '18 GHz'
fsl.res_bandwidth = "AUTO"
fsl.video_bandwidth = "AUTO"
fsl.sweep_time = "AUTO"

# Perform a sweep on full span, set the marker to the peak and some to that marker
fsl.single_sweep()
```

(continues on next page)

(continued from previous page)

```
m1.to_peak()
m1.zoom('20 MHz')

# take data from the zoomed-in region
fsl.single_sweep()
x, y = fsl.read_trace()
```

Bases: SCPIMixin, Instrument

Represents a Rohde&Schwarz FS Series of spectrum analyzers like FSL and FSW.

All physical values that can be set can either be as a string of a value and a unit (e.g. "1.2 GHz") or as a float value in the base units (Hz, dBm, etc.).

### property attenuation

Control attenuation in dB.

#### continue\_single\_sweep()

Continue with single sweep with synchronization.

### property continuous\_sweep\_enabled

Control continuous (True) or single sweep (False)

```
create_marker(num=1, is_delta_marker=False)
```

Create a marker.

### **Parameters**

- **num** The marker number (1-4)
- **is\_delta\_marker** True if the marker is a delta marker, default is False.

#### **Returns**

The marker object.

### property freq\_center

Control center frequency in Hz.

### property freq\_span

Control frequency span in Hz.

#### property freq\_start

Control start frequency in Hz.

#### property freq\_stop

Control stop frequency in Hz.

## property res\_bandwidth

Control resolution bandwidth in Hz. Can be set to 'AUTO'

#### single\_sweep()

Perform a single sweep with synchronization.

### property sweep\_time

Control sweep time in s. Can be set to 'AUTO'.

#### property trace\_mode

Control trace mode (Write: 'WRIT', Max Hold: 'MAXH', Min Hold: 'MINH', Average: 'AVER' or View: 'VIEW')

### property video\_bandwidth

Control video bandwidth in Hz. Can be set to 'AUTO'

Bases: FSSeries

Represents a Rohde&Schwarz FSL spectrum analyzer.

All physical values that can be set can either be as a string of a value and a unit (e.g. "1.2 GHz") or as a float value in the base units (Hz, dBm, etc.).

read\_trace(n\_trace=1)

Read trace data.

#### **Parameters**

**n\_trace** – The trace number (1-6). Default is 1.

#### Returns

2d numpy array of the trace data, [[frequency], [amplitude]].

Bases: FSSeries

Represents a Rohde&Schwarz FSW spectrum analyzer.

All physical values that can be set can either be as a string of a value and a unit (e.g. "1.2 GHz") or as a float value in the base units (Hz, dBm, etc.).

#### property activate\_channel

Control the name of the active channel. Note: The channel needs to be open on the device!

#### Returns

active channel name

### Return type

string

### property active\_channel

Control the name of the active channel. Note: The channel needs to be open on the device!

### Returns

active channel name

#### Return type

string

auto\_all()

Adjust all determinable settings automatically.

### auto\_freq()

Adjust center frequency automatically.

### auto\_level()

Adjust reference level automatically.

### property available\_channels

Measure open channel names and corresponding types

### create\_channel(channel\_type, channel\_name)

Create a new channel.

#### **Parameters**

- channel\_type Type of channel to be created. For example "PNOISE" or "SANA-LYZER"
- **channel\_name** Name of the channel to be added.

### delete\_channel(channel\_name)

Deletes an active channel.

### property nominal\_level

Control the nominal level of the instrument

```
read_trace(n_trace=1)
```

Read trace data of the active trace.

#### **Parameters**

**n\_trace** – The trace number (1-6). Default is 1.

#### Returns

2d numpy array of the trace data, [[frequency], [amplitude]].

### rename\_channel(current\_name, new\_name)

Rename current\_name of a channel to a new\_name.

### **Parameters**

- current\_name Channel to be renamed
- new\_name New name of the channel

### select\_channel(channel\_name)

Select an open channel

### **Parameters**

**channel\_name** – Channel to be selected.

### property split\_view

Control the viewmode of the device: True for split view or False for single channel view

## 7.51.3 R&S HMP4040 Power Supply

```
class pymeasure.instruments.rohdeschwarz.hmp.HMP4040(adapter, **kwargs)
```

Bases: SCPIMixin, Instrument

Represents a Rohde&Schwarz HMP4040 power supply.

#### beep()

Emit a single beep from the instrument.

### clear\_sequence(channel)

Clear the sequence of the selected channel.

#### property control\_method

Control manual front panel ('LOC'), remote ('REM') or manual/remote control('MIX') control or locks the front panel control ('RWL').

#### property current

Control output current in A. Range depends on instrument type.

### property current\_step

Control current step in A.

#### current\_to\_max()

Set current of the selected channel to its maximum value.

#### current\_to\_min()

Set current of the selected channel to its minimum value.

#### load\_sequence(slot)

Load a saved waveform from internal memory (slot 1, 2 or 3).

#### property max\_current

Get maximum current in A.

### property max\_voltage

Get maximum voltage in V.

### property measured\_current

Get current in A.

### property measured\_voltage

Get voltage in V.

#### property min\_current

Get minimum current in A.

### property min\_voltage

Get minimum voltage in V.

### property output\_enabled

Control the output on or off or check the output status.

#### property repetitions

Control umber of repetitions (0...255). If 0 is entered, the sequence isrepeated indefinitely.

#### save\_sequence(slot)

Save the sequence defined in the sequence property to internal memory (slot 1, 2 or 3).

### property selected\_channel

Control the selected channel.

### property selected\_channel\_active

Control the selected channel to active or inactive or check its status.

#### property sequence

Set sequence of triplets of voltage (V), current (A) and dwell time (s).

### set\_channel\_state(channel, state)

Set the state of the channel to active or inactive.

#### **Parameters**

- **channel** (*int*) Channel number to set the state of.
- **state** (*bool*) State of the channel, i.e. True for active, False for inactive.

### start\_sequence(channel)

Start the sequence of the selected channel.

### step\_current\_down()

Decreases current by one step.

## step\_current\_up()

Increase current by one step.

### step\_voltage\_down()

Decrease voltage by one step.

### step\_voltage\_up()

Increase voltage by one step.

### stop\_sequence(channel)

Stop the sequence defined in the sequence property of the selected channel.

### transfer\_sequence(channel)

Transfer the sequence defined in the sequence property to the selected channel.

## property version

Get the SCPI version the instrument's command set complies with.

### property voltage

Control output voltage in V. Increment 0.001 V.

## property voltage\_and\_current

Control output voltage (V) and current (A).

#### property voltage\_step

Control voltage step in V. Default 1 V.

#### voltage\_to\_max()

Set voltage of the selected channel to its maximum value.

### voltage\_to\_min()

Set voltage of the selected channel to its minimum value.

## 7.52 Santec

This section contains specific documentation on the Santec instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.52.1 Santec TSL-570 Tunable Laser

pymeasure.instruments.santec.ts1570

 $a lias \ of < module \ `pymeasure.instruments.santec. ts 1570' \ from \ `'home/docs/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkouts/readthedocs.org/user\_builds/pymeasure/checkout$ 

# 7.53 Siglent Technologies

This section contains specific documentation on the Siglent Technologies instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.53.1 Siglent Technologies Base Class

class pymeasure.instruments.siglenttechnologies.siglent\_spdbase.SPDBase(adapter,

name='Siglent SPDxxxxX instrument Base Class', \*\*kwargs)

Bases: SCPIUnknownMixin, Instrument

The base class for Siglent SPDxxxxX instruments.

Uses SPDChannel for measurement channels.

enable\_local\_interface(enable: bool = True)

Configure the availability of the local interface.

**Type** 

bool True: enables the local interface False: disables it.

#### property error

Get the error code and information of the instrument.

Туре

string

### property fw\_version

Get the software version of the instrument.

**Type** 

string

### recall\_config(index)

Recall a config from memory.

**Parameters** 

index – int: index of the location from which to recall the configuration

save\_config(index)

Save the current config to memory.

### **Parameters**

index – int: index of the location to save the configuration

```
property selected_channel
           Control the selected channel of the instrument.
           :type: int (dynamic)
     shutdown()
           Ensure that the voltage is turned to zero and disable the output.
     property system_status_code
           Get the system status register.
                Type
                    SystemStatusCode
class pymeasure.instruments.siglenttechnologies.siglent_spdbase.SPDSingleChannelBase(adapter,
                                                                                                      name='Siglent
                                                                                                      SPDxxxxX
                                                                                                      in-
                                                                                                      stru-
                                                                                                      ment
                                                                                                      Base
                                                                                                      Class',
                                                                                                      **kwargs)
     Bases: SPDBase
     enable_4W_mode(enable: bool = True)
           Enable 4-wire mode.
                Type
                    bool True: enables 4-wire mode False: disables it.
class pymeasure.instruments.siglenttechnologies.siglent_spdbase.SPDChannel(parent, id,
                                                                                          voltage_range: list
                                                                                          = [0, 16],
                                                                                          current_range: list
                                                                                          = [0, 8]
     Bases: Channel
     The channel class for Siglent SPDxxxxX instruments.
     configure_timer(step, voltage, current, duration)
           Configure the timer step.
                Parameters
                      • step – int: index of the step to save the configuration
                      • voltage – float: voltage setpoint of the step
                      • current – float: current limit of the step
                      • duration – int: duration of the step in seconds
     property current
           Measure the channel output current.
                Type
                    float
```

### property current\_limit

Control the output current configuration of the channel.

:type: float (dynamic)

### enable\_output(enable: bool = True)

Enable the channel output.

**Type** 

bool True: enables the output False: disables it

enable\_timer(enable: bool = True)

Enable the channel timer.

Type

bool True: enables the timer False: disables it

### property power

Measure the channel output power.

**Type** 

float

### property voltage

Measure the channel output voltage.

**Type** 

float

### property voltage\_setpoint

Control the output voltage configuration of the channel.

:type : float (dynamic)

class pymeasure.instruments.siglenttechnologies.siglent\_spdbase.SystemStatusCode(value,

names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

System status enums based on IntFlag

Used in conjunction with system\_status\_code.

Value	Enum
256	WAVEFORM_DISPLAY
64	TIMER_ENABLED
32	FOUR_WIRE
16	OUTPUT_ENABLED
1	CONSTANT_CURRENT
0	CONSTANT_VOLTAGE

## 7.53.2 Siglent SPD1168X Power Supply

Bases: SPDSingleChannelBase

Represent the Siglent SPD1168X Power Supply.

ch 1

#### Channel

SPDChannel

## 7.53.3 Siglent SPD1305X Power Supply

Bases: SPDSingleChannelBase

Represent the Siglent SPD1305X Power Supply.

ch\_1

#### Channel

SPDChannel

## 7.53.4 Siglent SDS1072CML Oscilloscope

Bases: SCPIMixin, Instrument

### Represents the SIGLENT SDS1072CML Oscilloscope

#### channel\_1

#### Channel

VoltageChannel

 $channel_2$ 

#### Channel

**VoltageChannel** 

trigger

#### Channel

TriggerChannel

arm()

Change the acquisition mode from 'STOPPED' to 'SINGLE'. Returns True if the scope is ready and armed.

### property internal\_state

Get the scope's Internal state change register and clears it.

#### property is\_ready

Get whether the scope is ready for the next acquisition (bool)

#### property status

Get the sampling status of the scope (Stop, Ready, Trig'd, Armed)

#### property template

Get a copy of the template that describes the various logical entities making up a complete waveform. In particular, the template describes in full detail the variables contained in the descriptor part of a waveform.

#### property time\_division

Control the time division to the closest possible value, rounding downwards, in seconds.

#### wait(time)

Stop the scope from doing anything until it has completed the current acquisition (p.146)

#### **Parameters**

time – time in seconds to wait for

### property waveform\_setup

Control the amount of data in a waveform to be transmitted to the controller with a dict containing the keys: - sparsing: The interval between data points (0 and 1 are all points, 4 are 1 out of 4 points, etc...) - number: the maximal number of data points to return. 0 is all points - first: the index of the data point from which to begin data transfer (0 is first)

 $\textbf{class} \ \, \textbf{pymeasure.instruments.siglenttechnologies.siglent\_sds1072cml.VoltageChannel} (\textit{parent}, \\ \textit{id})$ 

Bases: Channel

### Implementation of a SIGLENT SDS1072CML Oscilloscope channel

### property coupling

Control the channel coupling mode.(DC or AC)

### get\_descriptor(descriptor=None)

Get the descriptor of data being sent when querying device for waveform Will decode a descriptor if provided one (Raw byte stream), otherwise it will query it :return: dict: A dictionary with the keys: - numDataPoints: the number of points in the waveform - verticalGain: the voltage increment per code value (in V) - verticalOffset: the voltage offset to add to the decoded voltage values - horizInterval: the time interval between points in s - horizOffset:the offset to add to the time steps - descriptorOffset: Length of the C1:WF ALL,#9000000346 message

#### get\_waveform()

Get the waveforms displayed in the channel as a tuple with elements: - time: (1d array) the time in seconds since the trigger epoch for every voltage value in the waveforms - voltages: (1d array) the waveform in V

### property vertical\_division

Control the vertical sensitivity of a channel in V/divisions.

Bases: Channel

### Implementation of trigger control channel

### get\_trigger\_config()

Get the current trigger configuration as a dict with keys: - "type": condition that will trigger the acquisition of waveforms [EDGE, slew,GLIT,intv,runt,drop] - "source": trigger source (str, {EX,EX/5,C1,C2}) - "hold\_type": hold type (refer to page 131 of programing guide) - "hold\_value1": hold value1 (refer to page 131 of programing guide) - "level": Level at which the trigger will be set (float) - "slope": (str,{POS,NEG,WINDOW}) Triggers on rising, falling or Window. - "mode": behavior of the trigger following a triggering event (str, {NORM, AUTO, SINGLE,STOP}) - "coupling": (str,{AC,DC}) Coupling to the trigger channel

and updates the internal configuration status

## set\_trigger\_config(\*\*kwargs)

Set the current trigger configuration with keys: - "type": condition that will trigger the acquisition of waveforms [EDGE, slew,GLIT,intv,runt,drop] - "source": trigger source (str, {EX,EX/5,C1,C2}) - "hold\_type": hold type (refer to page 131 of programing guide) - "hold\_value1": hold value1 (refer to page 131 of programing guide) - "level": Level at which the trigger will be set (float) - "slope": (str,{POS,NEG,WINDOW}) Triggers on rising, falling or Window. - "mode": behavior of the trigger following a triggering event (str, {NORM, AUTO, SINGLE,STOP}) - "coupling": (str,{AC,DC}) Coupling to the trigger channel

Returns a flag indicating if all specified entries were correctly set on the oscilloscope and updates the interal trigger configuration

### property trigger\_coupling

Get the current trigger coupling as a dict with keys: - "source": trigger source whose coupling will be changed (str,  $\{EX,EX/5,C1,C2\}$ ) - "coupling": (str, $\{AC,DC\}$ ) Coupling to the trigger channel

#### property trigger\_level

Get the current trigger level as a dict with keys: - "source": trigger source whose level will be changed (str, {EX,EX/5,C1,C2}) - "level": Level at which the trigger will be set (float)

## property trigger\_mode

Get the current trigger mode as a dict with keys: - "mode": behavior of the trigger following a triggering event (str, {NORM, AUTO, SINGLE,STOP})

### property trigger\_setup

Get the current trigger setup as a dict with keys: -"type": condition that will trigger the acquisition of waveforms [EDGE, slew,GLIT,intv,runt,drop] - "source": trigger source (str, {EX,EX/5,C1,C2}) - "hold\_type": hold type (refer to page 131 of programing guide) - "hold\_value1": hold value1 (refer to page 131 of programing guide)

#### property trigger\_slope

Get the current trigger slope as a dict with keys: - "source": trigger source whose level will be changed (str, {EX,EX/5,C1,C2}) - "slope": (str, {POS,NEG,WINDOW}) Triggers on rising, falling or Window.

# 7.54 Signal Recovery

This section contains specific documentation on the Signal Recovery instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.54.1 DSP 7225 Lock-in Amplifier

Bases: DSPBase

Represents the Signal Recovery DSP 7225 lock-in amplifier.

Class inherits commands from the DSPBase parent class and utilizes dynamic properties for various properties.

```
lockin7225 = DSP7225("GPIB0::12::INSTR")
lockin7225.imode = "voltage mode"  # Set to measure voltages
lockin7225.reference = "internal"  # Use internal oscillator
lockin7225.fet = 1
                                         # Use FET pre-amp
lockin7225.shield = 0
                                        # Ground shields
lockin7225.coupling = 0
lockin7225.time_constant = 0.10
lockin7225.sensitivity = 2E-3
                                        # AC input coupling
                                       # Filter time set to 100 ms
                                        # Sensitivity set to 2 mV
                                        # Set oscillator frequency to 100 Hz
lockin7225.voltage = 1
                                         # Set oscillator amplitude to 1 V
lockin7225.gain = 20
                                         # Set AC gain to 20 dB
print(lockin7225.x)
                                          # Measure X channel voltage
                                         # Instrument shutdown
lockin7225.shutdown()
```

#### property adc1

Measure the voltage of the ADC1 input on the rear panel.

Returned value is a floating point number in volts.

### property adc2

Measure the voltage of the ADC2 input on the rear panel.

Returned value is a floating point number in volts.

## property auto\_gain

Control lock-in amplifier for automatic AC gain.

#### auto\_phase()

Adjusts the reference absolute phase to maximize the X channel output and minimize the Y channel output signals.

### auto\_sensitivity()

Adjusts the full-scale sensitivity so signal's magnitude lies between 30 - 90 % of full-scale.

```
buffer_to_float(buffer_data, sensitivity=None, sensitivity2=None, raise_error=True)
```

Converts fixed-point buffer data to floating point data.

The provided data is converted as much as possible, but there are some requirements to the data if all provided columns are to be converted; if a key in the provided data cannot be converted it will be omitted in the returned data or an exception will be raised, depending on the value of raise\_error.

The requirements for converting the data are as follows:

- Converting X, Y, magnitude and noise requires sensitivity data, which can either be part of the provided data or can be provided via the sensitivity argument
- The same holds for X2, Y2 and magnitude2 with sensitivity2.
- Converting the frequency requires both 'frequency part 1' and 'frequency part 2'.

#### **Parameters**

- **buffer\_data** (*dict*) The data to be converted. Must be in the format as returned by the *get\_buffer* method: a dict of numpy arrays.
- **sensitivity** If provided, the sensitivity used to convert X, Y, magnitude and noise. Can be provided as a float or as an array that matches the length of elements in *buffer\_data*. If both a sensitivity is provided and present in the buffer\_data, the provided value is used for the conversion, but the sensitivity in the buffer\_data is stored in the returned dict.
- **sensitivity2** Same as the first sensitivity argument, but for X2, Y2, magnitude2 and noise2.
- raise\_error (bool) Determines whether an exception is raised in case not all keys provided in buffer\_data can be converted. If False, the columns that cannot be converted are omitted in the returned dict.

#### Returns

Floating-point buffer data

### Return type

dict

### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### clear()

Clears the instrument status byte

#### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

### property coupling

Control the input coupling mode.

Valid values are 0 for AC coupling mode or 1 for DC coupling mode.

### property curve\_buffer\_bits

Control which data outputs are stored in the curve buffer.

Valid values are values are integers between 1 and 65,535 (or 2,097,151 in dual reference mode). (dynamic)

### property curve\_buffer\_interval

Control the time interval between the collection of successive points in the curve buffer.

Valid values to the time interval are integers in ms with a resolution of 5 ms; input values are rounded up to a multiple of 5. Valid values are values between 0 and 1,000,000,000 (corresponding to 12 days). The interval may be set to 0, which sets the rate of data storage to the curve buffer to 1.25 ms/point (800 Hz). However this only allows storage of the X and Y channel outputs. There is no need to issue a CBD 3 command to set this up since it happens automatically when acquisition starts.

### property curve\_buffer\_length

Control the length of the curve buffer.

Valid values are integers between 1 and 32,768, but the actual maximum amount of points is determined by the amount of curves that are stored, as set via the curve\_buffer\_bits property (32,768 / n).

#### property curve\_buffer\_status

Measure the status of the curve buffer acquisition.

Command returns four values: **First value - Curve Acquisition Status:** Number with 5 possibilities: 0: no activity 1: acquisition via TD command running 2: acquisition by a TDC command running 5: acquisition via TD command halted 6: acquisition bia TDC command halted **Second value - Number of Sweeps Acquired:** Number of sweeps already acquired. **Third value - Status Byte:** Decimal representation of the status byte (the same response as the ST command **Fourth value - Number of Points Acquired:** Number of points acquired in the curve buffer.

### property dac1

Control the voltage of the DAC1 output on the rear panel.

Valid values are floating point numbers between -12 to 12 V.

#### property dac2

Control the voltage of the DAC2 output on the rear panel.

Valid values are floating point numbers between -12 to 12 V.

#### property fet

Control the voltage preamplifier transistor type.

Valid values are 0 for bipolar or 1 for FET.

### property frequency

Control the oscillator frequency.

Valid values are floating point numbers representing the frequency in Hz. (dynamic)

#### property gain

Control the AC gain of signal channel amplifier.

### get\_buffer(quantity=None, convert\_to\_float=True, wait\_for\_buffer=True)

Retrieves the buffer after it has been filled. The data retrieved from the lock-in is in a fixed-point format, which requires translation before it can be interpreted as meaningful data. When *convert\_to\_float* is True the conversion is performed (if possible) before returning the data.

#### **Parameters**

- **quantity** (str) If provided, names the quantity that is to be retrieved from the curve buffer; can be any of: 'x', 'y', 'magnitude', 'phase', 'sensitivity', 'adc1', 'adc2', 'adc3', 'dac1', 'dac2', 'noise', 'ratio', 'log ratio', 'event', 'frequency part 1' and 'frequency part 2'; for both dual modes, additional options are: 'x2', 'y2', 'magnitude2', 'phase2', 'sensitivity2'. If no quantity is provided, all available data is retrieved.
- **convert\_to\_float** (*boo1*) Bool that determines whether to convert the fixed-point buffer-data to meaningful floating point values via the *buffer\_to\_float* method. If True, this method tries to convert all the available data to meaningful values; if this is not possible, an exception will be raised. If False, this conversion is not performed and the raw buffer-data is returned.
- wait\_for\_buffer (boo1) Bool that determines whether to wait for the data acquisition to finished if this method is called before the acquisition is finished. If True, the method waits until the buffer is filled before continuing; if False, the method raises an exception if the acquisition is not finished when the method is called.

#### property harmonic

Control the reference harmonic mode.

Valid values are integers. (dynamic)

### property id

Measure the model number of the instrument.

Returned value is an integer.

### property imode

Control the lock-in amplifier to detect a voltage or current signal.

Valid values are voltage mode, ``current mode, or low noise current mode.

#### init\_curve\_buffer()

Initializes the curve storage memory and status variables. All record of previously taken curves is removed.

#### property log\_ratio

Measure the log (base 10) of the ratio between the X channel and ADC1.

Returned value is a unitless floating point number equivalent to the mathematical expression log(X/ADC1).

### property mag

Measure the magnitude of the signal.

Returned value is a floating point number in volts.

#### property next\_error

Get the next error of the instrument (tuple of code and message).

## property options

Get the device options installed.

#### property phase

Measure the signal's absolute phase angle.

Returned value is a floating point number in degrees.

#### property ratio

Measure the ratio between the X channel and ADC1.

Returned value is a unitless floating point number equivalent to the mathematical expression X/ADC1.

## read(\*\*kwargs)

Read the response and remove extra unicode character from instrument readings.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

#### **Returns bytes**

Bytes response of the instrument (including termination).

### property reference

Control the oscillator reference input mode.

Valid values are internal, external rear or external front.

### property reference\_phase

Control the reference absolute phase angle.

Valid values are floating point numbers between 0 - 360 degrees.

### reset()

Resets the instrument.

### property sensitivity

Control the signal's measurement sensitivity range.

When in voltage measurement mode, valid values are discrete values from 2 nV to 1 V. When in current measurement mode, valid values are discrete values from 2 fA to 1  $\mu$ A (for normal current mode) or up to 10 nA (for low noise current mode).

### setChannelAMode()

Sets lock-in amplifier to measure a voltage signal only from the A input connector.

#### setDifferentialMode(lineFiltering=True)

Sets lock-in amplifier to differential mode, measuring A-B.

#### **set\_buffer**(points, quantities=None, interval=0.01)

Prepares the curve buffer for a measurement.

### **Parameters**

• **points** (*int*) – Number of points to be recorded in the curve buffer

- quantities (list) List containing the quantities (strings) that are to be recorded in the curve buffer, can be any of: 'x', 'y', 'magnitude', 'phase', 'sensitivity', 'adc1', 'adc2', 'adc3', 'dac1', 'dac2', 'noise', 'ratio', 'log ratio', 'event', 'frequency' (or 'frequency part 1' and 'frequency part 2'); for both dual modes, additional options are: 'x2', 'y2', 'magnitude2', 'phase2', 'sensitivity2'. Default is 'x' and 'y'.
- **interval** (*float*) The interval between two subsequent points stored in the curve buffer in s. Default is 10 ms.

### set\_voltage\_mode()

Sets lock-in amplifier to measure a voltage signal.

#### property shield

Control the input connector shield state.

Valid values are 0 to have shields grounded or 1 to have the shields floating (i.e., connected to ground via a 1 kOhm resistor).

#### shutdown()

Safely shutdown the lock-in amplifier.

Sets oscillator amplitude to 0 V and AC gain to 0 dB.

### property slope

Control the low-pass filter roll-off.

Valid values are the integers 6, 12, 18, or 24, which represents the slope of the low-pass filter in dB/octave.

#### start\_buffer()

Initiates data acquisition. Acquisition starts at the current position in the curve buffer and continues at the rate set by the STR command until the buffer is full.

### property status

Get the status byte and Master Summary Status bit.

#### property time\_constant

Control the filter time constant.

Valid values are a strict set of time constants from 10 us to 50,000 s. Returned values are floating point numbers in seconds.

## property voltage

Control the oscillator amplitude.

Valid values are floating point numbers between 0 to 5 V.

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

#### wait\_for\_buffer(timeout=None, delay=0.1)

Method that waits until the curve buffer is filled

### write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

#### **Parameters**

• command – command string to be sent to the instrument

• **kwargs** – Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

#### property x

Measure the output signal's X channel.

Returned value is a floating point number in volts.

### property xy

Measure both the X and Y channels.

Returned values are floating point numbers in volts.

### property y

Measure the output signal's Y channel.

Returned value is a floating point number in volts.

## 7.54.2 DSP 7265 Lock-in Amplifier

Bases: DSPBase

Represents the Signal Recovery DSP 7265 lock-in amplifier.

Class inherits commands from the DSPBase parent class and utilizes dynamic properties for various properties and includes additional functionality.

```
lockin7265 = DSP7265("GPIB0::12::INSTR")
lockin7265.imode = "voltage mode"  # Set to measure voltages
lockin7265.reference = "internal"  # Use internal oscillator
lockin7265.fet = 1  # Use FET pre-amp
lockin7265.fet = 1
                                                # Use FET pre-amp
lockin7265.shield = 0
                                                 # Ground shields
lockin7265.coupling = 0
                                                 # AC input coupling
lockin7265.time_constant = 0.10  # Filter time set to 100 ms lockin7265.sensitivity = 2E-3  # Sensitivity set to 2 mV
lockin7265.frequency = 100
                                                 # Set oscillator frequency to 100 Hz
                                                  # Set oscillator amplitude to 1 V
lockin7265.voltage = 1
lockin7265.gain = 20
                                                  # Set AC gain to 20 dB
print(lockin7265.x)
                                                  # Measure X channel voltage
                                                  # Instrument shutdown
lockin7265.shutdown()
```

#### property adc1

Measure the voltage of the ADC1 input on the rear panel.

Returned value is a floating point number in volts.

#### property adc2

Measure the voltage of the ADC2 input on the rear panel.

Returned value is a floating point number in volts.

## property adc3

Measure the ADC3 input voltage.

### property adc3\_time

Control the ADC3 sample time in seconds.

### property auto\_gain

Control lock-in amplifier for automatic AC gain.

### auto\_phase()

Adjusts the reference absolute phase to maximize the X channel output and minimize the Y channel output signals.

#### auto\_sensitivity()

Adjusts the full-scale sensitivity so signal's magnitude lies between 30 - 90 % of full-scale.

**buffer\_to\_float**(buffer\_data, sensitivity=None, sensitivity2=None, raise\_error=True)

Converts fixed-point buffer data to floating point data.

The provided data is converted as much as possible, but there are some requirements to the data if all provided columns are to be converted; if a key in the provided data cannot be converted it will be omitted in the returned data or an exception will be raised, depending on the value of raise\_error.

The requirements for converting the data are as follows:

- Converting X, Y, magnitude and noise requires sensitivity data, which can either be part of the provided data or can be provided via the sensitivity argument
- The same holds for X2, Y2 and magnitude2 with sensitivity2.
- Converting the frequency requires both 'frequency part 1' and 'frequency part 2'.

### **Parameters**

- **buffer\_data** (*dict*) The data to be converted. Must be in the format as returned by the *get\_buffer* method: a dict of numpy arrays.
- **sensitivity** If provided, the sensitivity used to convert X, Y, magnitude and noise. Can be provided as a float or as an array that matches the length of elements in *buffer\_data*. If both a sensitivity is provided and present in the buffer\_data, the provided value is used for the conversion, but the sensitivity in the buffer\_data is stored in the returned dict.
- **sensitivity2** Same as the first sensitivity argument, but for X2, Y2, magnitude2 and noise2.
- raise\_error (bool) Determines whether an exception is raised in case not all
  keys provided in buffer\_data can be converted. If False, the columns that cannot be
  converted are omitted in the returned dict.

### Returns

Floating-point buffer data

#### Return type

dict

### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### clear()

Clears the instrument status byte

## property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

#### property coupling

Control the input coupling mode.

Valid values are 0 for AC coupling mode or 1 for DC coupling mode.

#### property curve\_buffer\_bits

Control which data outputs are stored in the curve buffer.

Valid values are values are integers between 1 and 65,535 (or 2,097,151 in dual reference mode). (dynamic)

### property curve\_buffer\_interval

Control the time interval between the collection of successive points in the curve buffer.

Valid values to the time interval are integers in ms with a resolution of 5 ms; input values are rounded up to a multiple of 5. Valid values are values between 0 and 1,000,000,000 (corresponding to 12 days). The interval may be set to 0, which sets the rate of data storage to the curve buffer to 1.25 ms/point (800 Hz). However this only allows storage of the X and Y channel outputs. There is no need to issue a CBD 3 command to set this up since it happens automatically when acquisition starts.

#### property curve\_buffer\_length

Control the length of the curve buffer.

Valid values are integers between 1 and 32,768, but the actual maximum amount of points is determined by the amount of curves that are stored, as set via the curve\_buffer\_bits property (32,768 / n).

### property curve\_buffer\_status

Measure the status of the curve buffer acquisition.

Command returns four values: **First value - Curve Acquisition Status:** Number with 5 possibilities: 0: no activity 1: acquisition via TD command running 2: acquisition by a TDC command running 5: acquisition via TD command halted 6: acquisition bia TDC command halted **Second value - Number of Sweeps Acquired:** Number of sweeps already acquired. **Third value - Status Byte:** Decimal representation of the status byte (the same response as the ST command **Fourth value - Number of Points Acquired:** Number of points acquired in the curve buffer.

#### property dac1

Control the voltage of the DAC1 output on the rear panel.

Valid values are floating point numbers between -12 to 12 V.

### property dac2

Control the voltage of the DAC2 output on the rear panel.

Valid values are floating point numbers between -12 to 12 V.

### property dac3

Control the voltage of the DAC3 output on the rear panel.

Valid values are floating point numbers between -12 to 12 V.

### property dac4

Control the voltage of the DAC4 output on the rear panel.

Valid values are floating point numbers between -12 to 12 V.

#### property fet

Control the voltage preamplifier transistor type.

Valid values are 0 for bipolar or 1 for FET.

## property frequency

Control the oscillator frequency.

Valid values are floating point numbers representing the frequency in Hz. (dynamic)

#### property gain

Control the AC gain of signal channel amplifier.

### get\_buffer(quantity=None, convert\_to\_float=True, wait\_for\_buffer=True)

Retrieves the buffer after it has been filled. The data retrieved from the lock-in is in a fixed-point format, which requires translation before it can be interpreted as meaningful data. When *convert\_to\_float* is True the conversion is performed (if possible) before returning the data.

#### **Parameters**

• **quantity** (str) – If provided, names the quantity that is to be retrieved from the curve buffer; can be any of: 'x', 'y', 'magnitude', 'phase', 'sensitivity', 'adc1', 'adc2', 'adc3', 'dac1', 'dac2', 'noise', 'ratio', 'log ratio', 'event', 'frequency part 1' and 'frequency part 2'; for both dual modes, additional options are: 'x2', 'y2', 'magnitude2', 'phase2', 'sensitivity2'. If no quantity is provided, all available data is retrieved.

- **convert\_to\_float** (*bool*) Bool that determines whether to convert the fixed-point buffer-data to meaningful floating point values via the *buffer\_to\_float* method. If True, this method tries to convert all the available data to meaningful values; if this is not possible, an exception will be raised. If False, this conversion is not performed and the raw buffer-data is returned.
- wait\_for\_buffer (bool) Bool that determines whether to wait for the data acquisition to finished if this method is called before the acquisition is finished. If True, the method waits until the buffer is filled before continuing; if False, the method raises an exception if the acquisition is not finished when the method is called.

### property harmonic

Control the reference harmonic mode.

Valid values are integers. (dynamic)

#### property id

Measure the model number of the instrument.

Returned value is an integer.

#### property imode

Control the lock-in amplifier to detect a voltage or current signal.

Valid values are voltage mode, ``current mode, or low noise current mode.

#### init\_curve\_buffer()

Initializes the curve storage memory and status variables. All record of previously taken curves is removed.

### property log\_ratio

Measure the log (base 10) of the ratio between the X channel and ADC1.

Returned value is a unitless floating point number equivalent to the mathematical expression log(X/ADC1).

## property mag

Measure the magnitude of the signal.

Returned value is a floating point number in volts.

#### property next\_error

Get the next error of the instrument (tuple of code and message).

#### property options

Get the device options installed.

#### property phase

Measure the signal's absolute phase angle.

Returned value is a floating point number in degrees.

#### property ratio

Measure the ratio between the X channel and ADC1.

Returned value is a unitless floating point number equivalent to the mathematical expression X/ADC1.

## read(\*\*kwargs)

Read the response and remove extra unicode character from instrument readings.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

#### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

#### **Returns bytes**

Bytes response of the instrument (including termination).

#### property reference

Control the oscillator reference input mode.

Valid values are internal, external rear or external front.

## property reference\_phase

Control the reference absolute phase angle.

Valid values are floating point numbers between 0 - 360 degrees.

#### reset()

Resets the instrument.

### property sensitivity

Control the signal's measurement sensitivity range.

When in voltage measurement mode, valid values are discrete values from 2 nV to 1 V. When in current measurement mode, valid values are discrete values from 2 fA to 1  $\mu$ A (for normal current mode) or up to 10 nA (for low noise current mode).

### setChannelAMode()

Sets lock-in amplifier to measure a voltage signal only from the A input connector.

#### setDifferentialMode(lineFiltering=True)

Sets lock-in amplifier to differential mode, measuring A-B.

#### **set\_buffer**(points, quantities=None, interval=0.01)

Prepares the curve buffer for a measurement.

#### **Parameters**

- **points** (*int*) Number of points to be recorded in the curve buffer
- quantities (list) List containing the quantities (strings) that are to be recorded in the curve buffer, can be any of: 'x', 'y', 'magnitude', 'phase', 'sensitivity', 'adc1', 'adc2', 'adc3', 'dac1', 'dac2', 'noise', 'ratio', 'log ratio', 'event', 'frequency' (or 'frequency part 1' and 'frequency part 2'); for both dual modes, additional options are: 'x2', 'y2', 'magnitude2', 'phase2', 'sensitivity2'. Default is 'x' and 'y'.
- **interval** (*float*) The interval between two subsequent points stored in the curve buffer in s. Default is 10 ms.

#### set\_voltage\_mode()

Sets lock-in amplifier to measure a voltage signal.

#### property shield

Control the input connector shield state.

Valid values are 0 to have shields grounded or 1 to have the shields floating (i.e., connected to ground via a 1 kOhm resistor).

#### shutdown()

Safely shutdown the lock-in amplifier.

Sets oscillator amplitude to 0 V and AC gain to 0 dB.

#### property slope

Control the low-pass filter roll-off.

Valid values are the integers 6, 12, 18, or 24, which represents the slope of the low-pass filter in dB/octave.

### start\_buffer()

Initiates data acquisition. Acquisition starts at the current position in the curve buffer and continues at the rate set by the STR command until the buffer is full.

#### property status

Get the status byte and Master Summary Status bit.

#### property time\_constant

Control the filter time constant.

Valid values are a strict set of time constants from 10 us to 50,000 s. Returned values are floating point numbers in seconds.

### property voltage

Control the oscillator amplitude.

Valid values are floating point numbers between 0 to 5 V.

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

### wait\_for\_buffer(timeout=None, delay=0.1)

Method that waits until the curve buffer is filled

#### write(command, \*\*kwargs)

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- command command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

## write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

### **Parameters**

- **command** Command to send.
- values The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

### property x

Measure the output signal's X channel.

Returned value is a floating point number in volts.

### property xy

Measure both the X and Y channels.

Returned values are floating point numbers in volts.

#### property y

Measure the output signal's Y channel.

Returned value is a floating point number in volts.

# 7.55 Spellman HV

This section contains specific documentation on the Spellman HV instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

## 7.55.1 Spellman XRV High Voltage Power Supply

Bases: Instrument

A class representing the Spellman XRV series high voltage power supplies.

#### filament

#### Channel

Filament

### unscaled

#### Channel

**UnscaledData** 

#### property analog\_monitor

Measure the analog monitor read backs.

### Returns

dict

### Dict keys

voltage, current, filament, voltage\_setpoint, current\_setpoint, limit,
preheat, anode\_current

(dynamic)

### property baudrate

Set the baud rate (int, strictly in 9600, 19200, 38400, 57600, 115200).

#### check\_set\_errors()

Check for errors after sending a command.

#### Raise

ValueError if response is not \$

### static checksum(string\_to\_check)

Calculate the checksum.

#### **Parameters**

**string\_to\_check** – string to calculate the checksum from

### The checksum is computed as follows:

- Add all the bytes before <CSUM>, except <STX>, into a 16 bit (or larger) word. The bytes are added as unsigned integers.
- Take the two's complement.
- Truncate the result down to the eight least significant bits.
- Clear the most significant bit (bit 7) of the resultant byte, (bitwise AND with 0x7F).
- Set the next most significant bit (bit 6) of the resultant byte (bitwise OR with 0x40).

Using this method, the checksum is always a number between 0x40 and 0x7F. The checksum can never be confused with the  $\langle STX \rangle$  or  $\langle ETX \rangle$  control characters, since these have non overlapping ASCII values.

#### property configuration

Get the power supply configuration.

#### Returns

dict

## Dict keys

#### property current\_setpoint

Control the current setpoint in Amps (float).(dynamic)

## property dsp

Get the DSP part number and version (list).

#### property errors

Get the power supply errors (enum).

### property fpga

Get the FPGA part number and version (list).

### property hv\_on\_timer

Get the HV On time in hours (float).

#### property output\_enabled

Control the high voltage output (bool).

## property power\_limits

Control the power limits in Watts (list of int).

index = 0: power limit for large filament index = 1: power limit for small filament

7.55. Spellman HV 565

#### Raise

ValueError if limit is out of range

#### read()

Read from the device and check for errors.

#### Raise

ConnectionError if response doesn't start with <STX> or checksum is incorrect. The checksum check is omitted for TCPIP connections.

#### property scaling

Get scaling factors and polarity.

#### Returns

dict

#### Dict kevs

voltage, current, polarity

voltage is in V, current is in A, polarity 0: uni-polar, polarity 1: bipolar

### set\_scaling()

Set the scaling factors.

Used to set the scaling factor for analog\_monitor, current\_setpoint, voltage, voltage\_setpoint and system\_voltages.

#### property status

Get the power supply status (StatusCode enum).

### property system\_voltages

Measure the system voltages in Volts.

#### Returns

dict

#### Dict keys

temperature, reserved, anode, cathode, ac\_line\_cathode, dc\_rail\_cathode,
ac\_line\_anode, dc\_rail\_anode, lvps\_pos, lvps\_neg

(dynamic)

### property temperature

Measure the system temperature in °C (float).

#### property voltage

Measure the output voltage in Volts (float).(dynamic)

### property voltage\_setpoint

Control the voltage setpoint in Volts (float).(dynamic)

### wait\_for(query\_delay=0)

Wait for some time.

### **Parameters**

**query\_delay** – override the global query\_delay.

### write(command)

Write to the instrument.

Adds  $\langle STX \rangle$  (0x02) in front and checksum +  $\langle ETX \rangle$  (0x03) at end of every command before sending it. The checksum is omitted for TCPIP connections.

### class pymeasure.instruments.spellmanhv.spellmanXRV.Filament(parent, id)

Bases: Channel

A class representing the functions for the filament of the x-ray tube.

### property enabled

Set the filament status (bool).

### property large\_size\_enabled

Set the large filament state (bool)

#### property limit

Control the filament limit setpoint (int, strictly from 0 to 4095).

### property preheat

Control the filament preheat setpoint (int, strictly from 0 to 4095).

### class pymeasure.instruments.spellmanhv.spellmanXRV.UnscaledData(parent, id)

Bases: Channel

A class to handle the unscaled raw data of the Spellman XRV power supplies.

### property analog\_monitor

Measure the analog monitor read backs.

#### Returns

dict of int from 0 to 4095

#### Dict keys

voltage, current, filament, voltage\_setpoint, current\_setpoint, limit,
preheat, anode\_current

voltage\_setpoint, current\_setpoint, limit and preheat are for local mode operation only. anode\_current is only valid for bipolar units.

### property current\_setpoint

Control the current setpoint (int, strictly from 0 to 4095).

### property lvps\_monitor

Measure the -15 V low voltage power supply (int from 0 to 4095)

#### property system\_voltages

Measure the system voltages.

#### Returns

dict of int from 0 to 4095

### Dict keys

temperature, reserved, anode, cathode, ac\_line\_cathode, dc\_rail\_cathode,
ac\_line\_anode, dc\_rail\_anode, lvps\_pos, lvps\_neg

#### property voltage

Measure the output voltage (int from 0 to 4095).

### property voltage\_setpoint

Control the voltage setpoint (int, strictly from 0 to 4095).

7.55. Spellman HV 567

# 7.56 Stanford Research Systems

This section contains specific documentation on the Stanford Research Systems (SRS) instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.56.1 LDC500 Series Laser-diode Controllers

```
class pymeasure.instruments.srs.ldc500series.LDC500Series(adapter, name='LDC500Series',
                                                                     **kwargs)
     Bases: SCPIMixin, Instrument
     Represents an SRS LDC500Series laser diode controller and provides a high-level interface for interacting with
     the instrument.
     1d
                Channel
                    LDC500SeriesLD
     pd
                Channel
                    LDC500SeriesPD
     tec
                Channel
                    LDC500SeriesTEC
     check_errors()
           Read all errors from the instrument.
                Returns
                    List of [last_execution_error, last_command_error]
     check_set_errors()
           Check for errors after having set a property, and raise an error if any are present.
     property last_command_error
           Get the last command error code. This also resets the execution error code to 0.
     property last_execution_error
           Get the last execution error code. This also resets the execution error code to 0.
     property next_error
           Get next error not implemented, raises: NotImplementedError
     property options
           Get options not implemented, raises NotImplementedError
class pymeasure.instruments.srs.ldc500series.LDC500SeriesLD(parent, id)
     Bases: Channel
     LDC500Series channel for control of the laser-diode (LD).
     property current
           Measure the laser diode current, in mA (float).
```

#### property current\_limit

Control the laser diode current limit, in mA (float).

The ouput current is clamped to never exceed current\_limit under any conditions. If current\_limit is reduced below current\_setpoint, current\_setpoint is "dragged" down with current\_limit.

#### property current\_range

Control the laser diode current range (str "HIGH" or "LOW").

The currents corresponding to current range depend on the model:

	LDC500	LDC501	LDC502
HIGH	100mA	500mA	2000mA
LOW	50mA	250mA	1000mA

### property current\_setpoint

Control the laser diode current setpoint, in mA, when mode == "CC" (float).

### property enabled

Control whether the laser diode current source is enabled (bool).

### property interlock\_closed

Get the status of the interlock, True if closed, False if open. Laser will only operate with a closed interlock.

### property mode

Control the laser diode control mode, (str "CC" or "CP").

If the laser is on when mode is changed, the controller performs a *bumpless transfer* to switch control modes on-the-fly.

If mode is changed from "CC" to "CP", the present value of the photodiode current (or equivalently optical power) is measured and the CP setpoint is set to this measurement. If the measured photodiode current or power exceed the setpoint limit, an error is thrown.

If the mode is changed from "CP" to "CC", the present value of the laser current is measured and the CC setpoint set to this measurement. Note that, by hardware design, the measured laser current can never exceed the setpoint limit.

#### property modulation\_bandwidth

Control the laser diode modulation bandwidth (str "HIGH" or "LOW").

The analog modulation input is DC coupled, with the -3 dB roll-off frequency dependent on mode and modulation\_bandwidth:

	CC	CP
LOW	10kHz	100Hz
HIGH	1.2mhz	5kHz

#### property modulation\_enabled

Control whether analog modulation of the laser diode is enabled (bool)

### property voltage

Measure the laser diode voltage, in V (float).

#### property voltage\_limit

Control the laser diode voltage limit, in V (float strictly in range 0.1 to 10). The current source turns off when the voltage limit is exceeded.

#### class pymeasure.instruments.srs.ldc500series.LDC500SeriesPD(parent, id)

Bases: Channel

LDC500Series channel for control of the photodiode (PD).

### property bias

Control the photodiode bias (float strictly in range 0 to 5).

#### calibrate(power)

Set the photodiode responsivity, responsivity, via a real time power measurement.

#### **Parmeters:**

power (float): Real time power measurement, in mW.

#### property current

Measure the photodiode current, in uA (float).

### property current\_limit

Control the photodiode current limit, in uA, when mode == "CP" and photodiode\_units == "uA" (float).

#### property current\_setpoint

Control the photodiode current setpoint, in uA, when mode == "CP" and  $photodiode\_units == "uA"$  (float).

#### property power

Measure the photodiode power, in mW (float).

### property power\_limit

Control the photodiode power limit, in mW, when mode == "CP" and photodiode\_units == "mW" (float).

If power\_limit is reduced below power\_setpoint, power\_setpoint is "dragged" down with power\_limit.

### property power\_setpoint

Control the photodiode optical power setpoint, in mW, when mode == "CP" and  $photodiode\_units == "mW" (float).$ 

### property responsivity

Control the photodiode responsivity, in uA/mW (float).

Changing responsivity, either directly or via calibrate, indirectly changes:

- current\_setpoint and current\_limit if units == "mW"
- power\_setpoint and power\_limit if units == "uA"

This is to maintain the relationships:

- current\_setpoint = power\_setpoint x responsivity
- current\_limit = power\_limit x responsivity

### property units

Control the CP photodiode units (str "mW" or "uA").

### class pymeasure.instruments.srs.ldc500series.LDC500SeriesTEC(parent, id)

Bases: Channel

LDC500Series channel for control of the thermo-electric-controller (TEC).

### check\_temperature\_stability(tolerance=0.1, period=10, points=64)

Determine whether the temperature is stable at the temperature setpoint over a specified period.

#### **Parameters**

- **tolerance** Maximum allowed deviation from temperature setpoint, in degrees Centigrade.
- **period** Time period over which stability is checked, in seconds.

#### Returns

True if stable, False otherwise.

# property current

Measure the TEC current, in A.

# property current\_limit

Control the TEC current limit, in A (float strictly in range -4.5 to 4.5).

If current\_limit is reduced below current\_setpoint, current\_setpoint is *dragged* down with current\_limit.

# property current\_setpoint

Control the TEC current setpoint, in A, when mode == "CC" (float).

# property enabled

Control whether the TEC current source is enabled (bool).

# property mode

Control the TEC control mode (str "CC" or "CT").

If the TEC is on when mode is changed, the controller performs a *bumpless transfer* to swich control modes on-the-fly.

If mode is changed from "CT" to "CC", the present value of the TEC current is measured, and the CC setpoint is set to this measurement.

If mode is changed from "CC" to "CT", the present value of the temperature sensor is measured, and the CT setpoint is set to this measurement.

#### property resistance\_high\_limit

Control the TEC high resistance limit, in k (float).

#### property resistance\_limits

Control the TEC resistance limits, in k (two-tuple of floats).

# property resistance\_low\_limit

Control the TEC low resistance limit, in k (float).

#### property resistance\_setpoint

Control the TEC resistance setpoint, in k (float).

# property temperature

Measure the TEC temperature, in °C (float).

# property temperature\_high\_limit

Control the TEC high temperature limit, in °C (float).

# property temperature\_limits

Control the TEC temperature limits, in °C (two-tuple of floats).

# property temperature\_low\_limit

Control the TEC low temperature limit, in °C (float).

### property temperature\_setpoint

Control the TEC temperature setpoint, in °C (float).

#### property thermometer\_raw

Measure the raw thermometer reading, k, V, or A depending on the sensor type (float).

### property thermometer\_type

Control the temperature sensor type (ThermometerType enum).

# property voltage

Measure the TEC voltage, in V (float).

# property voltage\_limit

Control the TEC voltage limit, in V (float strictly in range -8.5 to 8.5).

# 

Block the program, waiting for the temperature to stabilize at the temperature setpoint.

#### **Parameters**

- **tolerance** Maximum allowed deviation from temperature setpoint, in degrees Centigrade.
- **period** Time period over which stability is checked, in seconds.
- **should\_stop** Function that returns True to stop waiting.
- timeout Maximum waiting time, in seconds.

## **Returns**

True when stable, False if stopped by should\_stop.

# Raises

**TimeoutError** – If the temperature does not stabilize within the timeout period.

Bases: IntEnum

Enumerator of thermometer types supported by the SRS LDC500Series. (NTC - negative temperature coefficient; RTD - Resistance temperature detector)

# Members:

- NTC10UA: 10 µA excitation for NTC thermistor.
- NTC100UA: 100 µA excitation for NTC thermistor.
- NTC1MA: 1 mA excitation for NTC thermistor.

- NTCAUTO : Automatically detects optimal excitation for NTC.
- RTD: Typically Pt-100 or other PTC sensors
- LM335 : LM335 and compatible temperature-to-voltage transducers.
- AD590 : AD590 and compatible temperature-to-current transducers.

# 7.56.2 SR510 Lock-in Amplifier

Bases: Instrument

# property frequency

A float property representing the SR510 input reference frequency

### property output

A float property that represents the SR510 output voltage in Volts.

## property phase

A float property that represents the SR510 reference to input phase offset in degrees. Queries return values between -180 and 180 degrees. This property can be set with a range of values between -999 to 999 degrees. Set values are mapped internal in the lockin to -180 and 180 degrees.

#### property sensitivity

A float property that represents the SR510 sensitivity value. This property can be set.

#### property status

A string property representing the bits set within the SR510 status byte

#### property time\_constant

A float property that represents the SR510 PRE filter time constant. This property can be set.

# 7.56.3 SR570 Lock-in Amplifier

Bases: SCPIUnknownMixin, Instrument

# property bias\_enabled

Boolean that turns the bias on or off. Allowed values are: True (bias on) and False (bias off)

### property bias\_level

A floating point value in V that sets the bias voltage level of the amplifier, in the [-5V,+5V] limits. The values are up to 1 mV precision level.

#### blank\_front()

"Blanks the frontend output of the device

# clear\_overload()

"Reset the filter capacitors to clear an overload condition

#### disable\_bias()

Turns the bias voltage off

#### disable\_offset\_current()

"Disables the offset current

# enable\_bias()

Turns the bias voltage on

#### enable\_offset\_current()

"Enables the offset current

### property filter\_type

A string that sets the filter type. Allowed values are: ['6dB Highpass', '12dB Highpass', '6dB Bandpass', '6dB Lowpass', '12dB Lowpass', 'none']

# property front\_blanked

Boolean that blanks(True) or un-blanks (False) the front panel

#### property gain\_mode

A string that sets the gain mode. Allowed values are: ['Low Noise', 'High Bandwidth', 'Low Drift']

### property high\_freq

A floating point value that sets the highpass frequency of the amplifier, which takes a discrete value in a 1-3 sequence. Values are truncated to the closest allowed value if not exact. Allowed values range from 0.03 Hz to 1 MHz.

# property invert\_signal\_sign

An boolean sets the signal invert sense. Allowed values are: True (inverted) and False (not inverted).

# property low\_freq

A floating point value that sets the lowpass frequency of the amplifier, which takes a discrete value in a 1-3 sequence. Values are truncated to the closest allowed value if not exact. Allowed values range from 0.03 Hz to 1 MHz.

# property offset\_current

A floating point value in A that sets the absolute value of the offset current of the amplifier, in the [1pA,5mA] limits. The offset current takes discrete values in a 1-2-5 sequence. Values are truncated to the closest allowed value if not exact.

# property offset\_current\_enabled

Boolean that turns the offset current on or off. Allowed values are: True (current on) and False (current off).

#### property offset\_current\_sign

An string that sets the offset current sign. Allowed values are: 'positive' and 'negative'.

### property sensitivity

A floating point value that sets the sensitivity of the amplifier, which takes discrete values in a 1-2-5 sequence. Values are truncated to the closest allowed value if not exact. Allowed values range from 1 pA/V to 1 mA/V.

# property signal\_inverted

Boolean that inverts the signal if True

# unblank\_front()

Un-blanks the frontend output of the device

# 7.56.4 SR830 Lock-in Amplifier

Bases: Instrument

#### property adc1

Reads the Aux input 1 value in Volts with 1/3 mV resolution.

# property adc2

Reads the Aux input 2 value in Volts with 1/3 mV resolution.

## property adc3

Reads the Aux input 3 value in Volts with 1/3 mV resolution.

#### property adc4

Reads the Aux input 4 value in Volts with 1/3 mV resolution.

#### auto\_offset(channel)

Offsets the channel (X, Y, or R) to zero

# property aux\_in\_1

Reads the Aux input 1 value in Volts with 1/3 mV resolution.

#### property aux\_in\_2

Reads the Aux input 2 value in Volts with 1/3 mV resolution.

#### property aux\_in\_3

Reads the Aux input 3 value in Volts with 1/3 mV resolution.

# property aux\_in\_4

Reads the Aux input 4 value in Volts with 1/3 mV resolution.

# property aux\_out\_1

A floating point property that controls the output of Aux output 1 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

## property aux\_out\_2

A floating point property that controls the output of Aux output 2 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

#### property aux\_out\_3

A floating point property that controls the output of Aux output 3 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

#### property aux\_out\_4

A floating point property that controls the output of Aux output 4 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

# **buffer\_measure**(count, stopRequest=None, delay=0.001)

Start a fast measurement mode and transfers data from buffer to extract mean and std measurements

Return the mean and std from both channels

# property channel1

A string property that represents the type of Channel 1, taking the values X, R, X Noise, Aux In 1, or Aux In 2. This property can be set.

#### property channel2

A string property that represents the type of Channel 2, taking the values Y, Theta, Y Noise, Aux In 3, or Aux In 4. This property can be set.

#### clear()

Clear the instrument status byte.

# property dac1

A floating point property that controls the output of Aux output 1 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

# property dac2

A floating point property that controls the output of Aux output 2 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

## property dac3

A floating point property that controls the output of Aux output 3 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

## property dac4

A floating point property that controls the output of Aux output 4 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

# property err\_status

Reads the value of the lockin error (ERR) status byte. Returns an IntFlag type with positions within the string corresponding to different error flags:

Bit	Status	
0	unused	
1	backup error	
2	RAM error	
3	unused	
4	ROM error	
5	GPIB error	
6	DSP error	
7	DSP error	

**fill\_buffer**(count: int, has\_aborted=<function SR830.<lambda>>, delay=0.001)

Fill two numpy arrays with the content of the instrument buffer

Eventually waiting until the specified number of recording is done

# property filter\_slope

An integer property that controls the filter slope, which can take on the values 6, 12, 18, and 24 dB/octave. Values are truncated to the next highest level if they are not exact.

# property filter\_synchronous

A boolean property that controls the synchronous filter. This property can be set. Allowed values are: True or False

## property frequency

A floating point property that represents the lock-in frequency in Hz. This property can be set.

# get\_buffer(channel=1, start=0, end=None)

Acquires the 32 bit floating point data through binary transfer

### get\_scaling(channel)

Returns the offset percent and the expansion term that are used to scale the channel in question

# property harmonic

An integer property that controls the harmonic that is measured. Allowed values are 1 to 19999. Can be set.

# property id

Get the identification of the instrument.

# property input\_config

An string property that controls the input configuration. Allowed values are: ['A', 'A - B', 'I (1 MOhm)', 'I (100 MOhm)']

# property input\_coupling

An string property that controls the input coupling. Allowed values are: ['AC', 'DC']

# property input\_grounding

An string property that controls the input shield grounding. Allowed values are: ['Float', 'Ground']

# property input\_notch\_config

An string property that controls the input line notch filter status. Allowed values are: ['None', 'Line', '2 x Line', 'Both']

# is\_out\_of\_range()

Returns True if the magnitude is out of range

# property lia\_status

Reads the value of the lockin amplifier (LIA) status byte. Returns a binary string with positions within the string corresponding to different status flags:

Bit	Status
0	Input/Amplifier overload
1	Time constant filter overload
2	Output overload
3	Reference unlock
4	Detection frequency range switched
5	Time constant changed indirectly
6	Data storage triggered
7	unused

# load\_setup(setup\_number: int)

Load a previously saved instrument configuration from the memory referred to by an integer

#### **Parameters**

**setup\_number** – the integer referring to the memory (between 1 and 9 (included))

#### property magnitude

Reads the magnitude in Volts.

# output\_conversion(channel)

Returns a function that can be used to determine the signal from the channel output (X, Y, or R)

# pause\_scan()

Pause the data recording

### property phase

A floating point property that represents the lock-in phase in degrees. This property can be set.

# quick\_range()

While the magnitude is out of range, increase the sensitivity by one setting

#### property reference\_source

An string property that controls the reference source. Allowed values are: ['External', 'Internal']

### property reference\_source\_trigger

A string property that controls the reference source triggering. Allowed values are: ['SINE', 'POS EDGE', 'NEG EDGE']

#### reset()

Reset the instrument.

## property sample\_frequency

Gets the sample frequency in Hz

### save\_setup(setup number: int)

Save the current instrument configuration (all parameters) in a memory referred to by an integer

#### **Parameters**

**setup\_number** – the integer referring to the memory (between 1 and 9 (included))

# property sensitivity

A floating point property that controls the sensitivity in Volts, which can take discrete values from 2 nV to 1 V. Values are truncated to the next highest level if they are not exact.

# set\_scaling(channel, precent, expand=0)

Sets the offset of a channel (X=1, Y=2, R=3) to a certain percent (-105% to 105%) of the signal, with an optional expansion term (0, 10=1, 100=2)

# property sine\_voltage

A floating point property that represents the reference sine-wave voltage in Volts. This property can be set.

```
snap(val1='X', val2='Y', *vals)
```

Method that records and retrieves 2 to 6 parameters at a single instant. The parameters can be one of: X, Y, R, Theta, Aux In 1, Aux In 2, Aux In 3, Aux In 4, Frequency, CH1, CH2. Default is "X" and "Y".

#### **Parameters**

- val1 first parameter to retrieve
- val2 second parameter to retrieve
- vals other parameters to retrieve (optional)

## start\_scan()

Start the data recording into the buffer

#### property status

Get the status byte and Master Summary Status bit.

# property theta

Reads the theta value in degrees.

#### property time\_constant

A floating point property that controls the time constant in seconds, which can take discrete values from 10 microseconds to 30,000 seconds. Values are truncated to the next highest level if they are not exact.

wait\_for\_buffer(count, has\_aborted=<function SR830.<lambda>>, timeout=60, timestep=0.01)

Wait for the buffer to fill a certain count

# property x

Reads the X value in Volts.

# property xy

Reads the X and Y values in Volts.

# property y

Reads the Y value in Volts.

# 7.56.5 SR860 Lock-in Amplifier

### Bases: Instrument

# property adc1

Reads the Aux input 1 value in Volts with 1/3 mV resolution.

# property adc2

Reads the Aux input 2 value in Volts with 1/3 mV resolution.

#### property adc3

Reads the Aux input 3 value in Volts with 1/3 mV resolution.

# property adc4

Reads the Aux input 4 value in Volts with 1/3 mV resolution.

# property aux\_in\_1

Reads the Aux input 1 value in Volts with 1/3 mV resolution.

#### property aux\_in\_2

Reads the Aux input 2 value in Volts with 1/3 mV resolution.

# property aux\_in\_3

Reads the Aux input 3 value in Volts with 1/3 mV resolution.

# property aux\_in\_4

Reads the Aux input 4 value in Volts with 1/3 mV resolution.

#### property aux\_out\_1

A floating point property that controls the output of Aux output 1 in Volts, taking values between -10.5 V and  $\pm$ 10.5 V. This property can be set.

## property aux\_out\_2

A floating point property that controls the output of Aux output 2 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

#### property aux\_out\_3

A floating point property that controls the output of Aux output 3 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

#### property aux\_out\_4

A floating point property that controls the output of Aux output 4 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

#### property dac1

A floating point property that controls the output of Aux output 1 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

### property dac2

A floating point property that controls the output of Aux output 2 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

## property dac3

A floating point property that controls the output of Aux output 3 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

# property dac4

A floating point property that controls the output of Aux output 4 in Volts, taking values between -10.5 V and +10.5 V. This property can be set.

# property dcmode

A string property that represents the sine out dc mode. This property can be set. Allowed values are:['COM', 'DIF', 'common', 'difference']

## property detectedfrequency

Returns the actual detected frequency in HZ.

# property extfreqency

Returns the external frequency in Hz.

#### property filer\_synchronous

Access filter\_synchronous attribute with filer\_synchronous (sic) property.

Deprecated since version 0.16.0: Use filter\_synchronous instead.

# property filter\_advanced

A string property that represents the advanced filter. This property can be set. Allowed values are:['Off', 'On']

#### property filter\_slope

A integer property that sets the filter slope to 6 dB/oct(i=0), 12 DB/oct(i=1), 18 dB/oct(i=2), 24 dB/oct(i=3).

# property filter\_synchronous

A string property that represents the synchronous filter. This property can be set. Allowed values are:['Off', 'On']

#### property frequency

A floating point property that represents the lock-in frequency in Hz. This property can be set.

# property frequencypreset1

A floating point property that represents the preset frequency for the F1 preset button. This property can be set.

### property frequencypreset2

A floating point property that represents the preset frequency for the F2 preset button. This property can be set

# property frequencypreset3

A floating point property that represents the preset frequency for the F3 preset button. This property can be set.

### property frequencypreset4

A floating point property that represents the preset frequency for the F4 preset button. This property can be set.

# property front\_panel

Turns the front panel blanking on(i=0) or off(i=1).

# property get\_noise\_bandwidth

Returns the equivalent noise bandwidth, in hertz.

# property get\_signal\_strength\_indicator

Returns the signal strength indicator.

## property gettimebase

Returns the current 10 MHz timebase source.

# property harmonic

An integer property that controls the harmonic that is measured. Allowed values are 1 to 99. Can be set.

## property harmonicdual

An integer property that controls the harmonic in dual reference mode that is measured. Allowed values are 1 to 99. Can be set.

# property horizontal\_time\_div

A integer property for the horizontal time/div according to the following table: ['0=0.5s', '1=1s', '2=2s', '3=5s', '4=10s', '5=30s', '6=1min', '7=2min', '8=5min', '9=10min', '10=30min', '11=1hour', '12=2hour', '13=6hour', '14=12hour', '15=1day', '16=2days']

#### property input\_config

A string property that represents the voltage input mode. This property can be set. Allowed values are:['A', 'A-B']

## property input\_coupling

A string property that represents the input coupling. This property can be set. Allowed values are:['AC', 'DC']

#### property input\_current\_gain

A string property that represents the current input gain. This property can be set. Allowed values are:['1MEG', '100MEG']

# property input\_grounding

A string property that represents the input shield grounding. This property can be set. Allowed values are:['Float', 'Ground']

# property input\_range

A string property that represents the input range. This property can be set. Allowed values are:['1V', '300M', '100M', '30M', '10M']

### property input\_shields

A string property that represents the input shield grounding. This property can be set. Allowed values are:['Float', 'Ground']

### property input\_signal

A string property that represents the signal input. This property can be set. Allowed values are:['VOLT', 'CURR', 'voltage', 'current']

### property input\_voltage\_mode

A string property that represents the voltage input mode. This property can be set. Allowed values are:['A', 'A-B']

# property internal frequency

A floating property that represents the internal lock-in frequency in Hz This property can be set.

# property magnitude

Reads the magnitude in Volts.

## property parameter\_DAT1

A integer property that assigns a parameter to data channel 1(green). This parameters can be set. Allowed values are:['i=', '0=Xoutput', '1=Youtput', '2=Routput', 'Thetaoutput', '4=Aux IN1', '5=Aux IN2', '6=Aux IN3', '7=Aux IN4', '8=Xnoise', '9=Ynoise', '10=AUXOut1', '11=AuxOut2', '12=Phase', '13=Sine Out amplitude', '14=DCLevel', '15I=nt.referenceFreq', '16=Ext.referenceFreq']

# property parameter\_DAT2

A integer property that assigns a parameter to data channel 2(blue). This parameters can be set. Allowed values are: ['i=', '0=Xoutput', '1=Youtput', '2=Routput', 'Thetaoutput', '4=Aux IN1', '5=Aux IN2', '6=Aux IN3', '7=Aux IN4', '8=Xnoise', '9=Ynoise', '10=AUXOut1', '11=AuxOut2', '12=Phase', '13=Sine Out amplitude', '14=DCLevel', '15I=nt.referenceFreq', '16=Ext.referenceFreq']

# property parameter\_DAT3

A integer property that assigns a parameter to data channel 3(yellow). This parameters can be set. Allowed values are:['i=', '0=Xoutput', '1=Youtput', '2=Routput', 'Thetaoutput', '4=Aux IN1', '5=Aux IN2', '6=Aux IN3', '7=Aux IN4', '8=Xnoise', '9=Ynoise', '10=AUXOut1', '11=AuxOut2', '12=Phase', '13=Sine Out amplitude', '14=DCLevel', '15I=nt.referenceFreq', '16=Ext.referenceFreq']

### property parameter\_DAT4

A integer property that assigns a parameter to data channel 3(orange). This parameters can be set. Allowed values are: ['i=', '0=Xoutput', '1=Youtput', '2=Routput', 'Thetaoutput', '4=Aux IN1', '5=Aux IN2', '6=Aux IN3', '7=Aux IN4', '8=Xnoise', '9=Ynoise', '10=AUXOut1', '11=AuxOut2', '12=Phase', '13=Sine Out amplitude', '14=DCLevel', '15I=nt.referenceFreq', '16=Ext.referenceFreq']

#### property phase

A floating point property that represents the lock-in phase in degrees. This property can be set.

# property reference\_externalinput

A string property that represents the external reference input. This property can be set. Allowed values are:['50OHMS', '1MEG']

#### property reference\_source

A string property that represents the reference source. This property can be set. Allowed values are:['INT', 'EXT', 'DUAL', 'CHOP']

# property reference\_source\_trigger

A string property that represents the external reference trigger mode. This property can be set. Allowed values are:['SIN', 'POS', 'NEG', 'POSTTL', 'NEGTTL']

### property reference\_triggermode

A string property that represents the external reference trigger mode. This property can be set. Allowed values are:['SIN', 'POS', 'NEG', 'POSTTL', 'NEGTTL']

### property screen\_layout

A integer property that Sets the screen layout to trend(i=0), full strip chart history(i=1), half strip chart history(i=2), full FFT(i=3), half FFT(i=4) or big numerical(i=5).

#### screenshot()

Take screenshot on device The DCAP command saves a screenshot to a USB memory stick. This command is the same as pressing the [Screen Shot] key. A USB memory stick must be present in the front panel USB port.

# property sensitivity

A floating point property that controls the sensitivity in Volts, which can take discrete values from 2 nV to 1 V. Values are truncated to the next highest level if they are not exact.

# property sensitvity

Access sensitivity attribute with sensitvity (sic) property.

Deprecated since version 0.16.0: Use sensitivity instead.

# property sine\_amplitudepreset1

Floating point property representing the preset sine out amplitude, for the A1 preset button. This property can be set.

## property sine\_amplitudepreset2

Floating point property representing the preset sine out amplitude, for the A2 preset button. This property can be set.

# property sine\_amplitudepreset3

Floating point property representing the preset sine out amplitude, for the A3 preset button. This property can be set.

### property sine\_amplitudepreset4

Floating point property representing the preset sine out amplitude, for the A3 preset button. This property can be set.

# property sine\_dclevelpreset1

A floating point property that represents the preset sine out dc level for the L1 button. This property can be set.

# property sine\_dclevelpreset2

A floating point property that represents the preset sine out dc level for the L2 button. This property can be set.

# property sine\_dclevelpreset3

A floating point property that represents the preset sine out dc level for the L3 button. This property can be set.

#### property sine\_dclevelpreset4

A floating point property that represents the preset sine out dc level for the L4 button. This property can be set.

# property sine\_voltage

A floating point property that represents the reference sine-wave voltage in Volts. This property can be set.

snap(val1='X', val2='Y', val3=None)

retrieve 2 or 3 parameters at once parameters can be chosen by index, or enumeration as follows:

index	enumeration	parameter
0	X	X output
1	Y	Y output
2	R	R output
3	THeta	output
4	IN1	Aux In1
5	IN2	Aux In2
6	IN3	Aux In3
7	IN4	Aux In4
8	XNOise	Xnoise
9	YNOise	Ynoise
10	OUT1	Aux Out1
11	OUT2	Aux Out2
12	PHAse	Reference Phase
13	SAMp	Sine Out Amplitude
14	LEVel	DC Level
15	FInt	Int. Ref. Frequency
16	FExt	Ext. Ref. Frequency

#### **Parameters**

- val1 parameter enumeration/index
- val2 parameter enumeration/index
- val3 parameter enumeration/index (optional)

# **Defaults:**

# snap\_all()

snap X,Y,R,THETA parameters at once

# property strip\_chart\_dat1

A integer property that turns the strip chart graph of data channel 1 off(i=0) or on(i=1).

## property strip\_chart\_dat2

A integer property that turns the strip chart graph of data channel 2 off(i=0) or on(i=1).

# property strip\_chart\_dat3

A integer property that turns the strip chart graph of data channel 1 off(i=0) or on(i=1).

# property strip\_chart\_dat4

A integer property that turns the strip chart graph of data channel 4 off(i=0) or on(i=1).

# property theta

Reads the theta value in degrees.

## property time\_constant

A floating point property that controls the time constant in seconds, which can take discrete values from 10 microseconds to 30,000 seconds. Values are truncated to the next highest level if they are not exact.

#### property timebase

Sets the external 10 MHZ timebase to auto(i=0) or internal(i=1).

# property x

Reads the X value in Volts

#### property y

Reads the Y value in Volts

# 7.57 T&C Power Conversion

This section contains specific documentation on the instruments from T&C Power Conversion that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

### 7.57.1 T&C Power Conversion AG Series Plasma Generator CXN

Bases: Instrument

T&C Power Conversion AG Series Plasma Generator CXN (also rebranded by AJA International Inc as 0113 GTC or 0313 GTC)

Connection to the device is made through an RS232 serial connection. The communication settings are fixed in the device at 38400, stopbit one, parity none. The device uses a command response system where every receipt of a command is acknowledged by returning a '\*'. A '?' is returned to indicates the command was not recognized by the device.

A command messages always consists of the following bytes (B): 1B - header (always 'C'), 1B - address (ignored), 2B - command id, 2B - parameter 1, 2B - parameter, 2B - checksum

A response message always consists of: 1B - header (always 'R'), 1B - address of the device, 2B - length of the data package, variable length data, 2B - checksum response messages are received after the acknowledge byte.

# **Parameters**

- adapter pyvisa resource name of the instrument or adapter instance
- name (string) Name of the instrument.
- **kwargs** Any valid key-word argument for Instrument

# **1** Note

In order to enable setting any parameters one has to request control and periodically (at least once per 2s) poll any value from the device. Failure to do so will mean loss of control and the device will reset certain parameters (setpoint, disable RF, ...). If no value should be polled but control should remain active one can also use the ping method.

#### preset\_1

# Channel

**PresetChannel** 

#### preset\_2

# Channel

**PresetChannel** 

#### preset\_3

#### Channel

**PresetChannel** 

# preset\_4

#### Channel

**PresetChannel** 

# preset\_5

# Channel

**PresetChannel** 

### preset\_6

# Channel

**PresetChannel** 

#### preset\_7

#### Channel

**PresetChannel** 

# preset\_8

## Channel

**PresetChannel** 

# preset\_9

#### Channel

**PresetChannel** 

**class Status**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntFlag

IntFlag type used to represent the CXN status.

The used bits correspond to: bit 14: Analog interface enabled, bit 11: Interlock open, bit 10: Over temperature, bit 9: Reverse power limit, bit 8: Forward power limit, bit 6: MCG mode active, bit 5: load power leveling active, bit 4, External RF source active, bit 0: RF power on.

# property dc\_voltage

Get the DC voltage in volts.

# property firmware\_version

Get the UI-processor and RF-processor firmware version numbers.

#### property frequency

Get operating frequency in Hz.

## property id

Get the device identification string.

### property load\_capacity

Control the percentage of full-scale value of the load capacity. It can be set only when manual\_mode is True.

# property manual\_mode

Control the manual tuner mode.

## property operation\_mode

Control the operation mode.

### ping()

Send a ping to the instrument.

# property power

Get power readings for forward/reverse/load power in watts.

#### property power\_limit

Get maximum power of the power supply.

### property preset\_slot

Control which preset slot will be used for auto-tune mode. Valid values are 0 to 9. 0 means no preset will be used

#### property pulse\_params

Get pulse on/off time of the pulse waveform.

# property ramp\_rate

Control the ramp rate in watts/second.

# property ramp\_start\_power

Control the ramp starting power in watts.

### read()

Reads a response message from the instrument.

This method determines the length of the message from the automatically by reading the message header and also checks for a correct checksum.

# Returns

the data fields

# Return type

bytes

#### **Raises**

ValueError - if a checksum error is detected

# release\_control()

Release instrument control.

This will reset certain properties to safe defaults and disable the RF output.

# request\_control()

Request control of the instrument.

This is required to be able to set any properties.

# property reverse\_power\_limit

Get maximum reverse power.

#### property rf\_enabled

Control the RF output.

# property serial

Get the serial number of the instrument.

## property setpoint

Control the setpoint power level in watts.

#### property status

Get status field. The return value is represented by the IntFlag type Status.

#### property temperature

Get heat sink temperature in deg Celsius.

### property tune\_capacity

Control the percentage of full-scale value of the tune capacity. It can be set only when manual\_mode is True.

### property tuner

Get type of the used tuner.

values(command, cast=<class 'int'>, separator=', ', preprocess\_reply=None, \*\*kwargs)

Write a command to the instrument and return a list of formatted values from the result.

This is derived from CommonBase.values and adapted here for use with bytes communication messages (no str conversion and strip). It is implemented as a general method to allow using it equally in PresetChannel and CXN. See Github issue #784 for details.

# **Parameters**

- command SCPI command to be sent to the instrument
- **separator** A separator character to split the string into a list
- $\bullet$  cast A type to cast the result
- **preprocess\_reply** optional callable used to preprocess values received from the instrument. The callable returns the processed string.

# Returns

A list of the desired type, or strings where the casting fails

## write(command)

Writes a command to the instrument and includes needed required header and address.

#### **Parameters**

**command** (str) – command to be sent to the instrument

class pymeasure.instruments.tcpowerconversion.tccxn.PresetChannel(parent, id)

Bases: Channel

## property load\_capacity

Control the percentage of full-scale value of the load capacity preset.

#### property tune\_capacity

Control the percentage of full-scale value of the tune capacity preset.

values(command, cast=<class 'int'>, separator=', ', preprocess\_reply=None, \*\*kwargs)

Write a command to the instrument and return a list of formatted values from the result.

This is derived from CommonBase.values and adapted here for use with bytes communication messages (no str conversion and strip). It is implemented as a general method to allow using it equally in PresetChannel and CXN. See Github issue #784 for details.

#### **Parameters**

- command SCPI command to be sent to the instrument
- **separator** A separator character to split the string into a list
- cast A type to cast the result
- **preprocess\_reply** optional callable used to preprocess values received from the instrument. The callable returns the processed string.

#### Returns

A list of the desired type, or strings where the casting fails

# 7.58 TDK Lambda

This section contains specific documentation on the TDK Lambda instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.58.1 TDK Lambda Genesys 40-38 DC power supply

Bases: TDK\_Lambda\_Base

Represents the TDK Lambda Genesys 40-38 DC power supply. Class inherits commands from the TDK\_Lambda\_Base parent class and utilizes dynamic properties adjust valid values on various properties.

```
psu = TDK_Gen40_38("COM3", 6)  # COM port and daisy-chain address
psu.remote = "REM"  # PSU in remote mode
psu.output_enabled = True  # Turn on output
psu.ramp_to_current(2.0)  # Ramp to 2.0 A of current
print(psu.current)  # Measure actual PSU current
print(psu.voltage)  # Measure actual PSU voltage
psu.shutdown()  # Run shutdown command
```

The initialization of a TDK instrument requires the current address of the TDK power supply. The default address for the TDK Lambda is 6.

# **Parameters**

- adapter VISAAdapter instance
- name Instrument name. Default is "TDK Lambda Gen40-38"
- address Serial port daisy chain number. Default is 6.

### property address

Set the address of the power supply.

Valid values are integers between 0 - 30 (inclusive).

7.58. TDK Lambda 589

### property auto\_restart\_enabled

Control the auto restart mode, which restores the power supply to the last output voltage and current settings with output enabled on startup.

Valid values are True to restore output settings with output enabled on startup and False to disable restoration of settings and output disabled on startup.

#### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

## check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

### check\_set\_errors()

Only use this command for setting commands, i.e. non-querying commands.

Any non-querying commands (i.e., a command that does NOT have the "?" symbol in it like the instrument command "PV 10") will automatically return an "OK" reply for valid command or an error code. This is done to confirm that the instrument has received the command. Any querying commands (i.e., a command that does have the "?" symbol in it like the instrument command "PV?") will return the requested value, not the confirmation.

### clear()

Clear FEVE and SEVE registers to zero.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property current

Measure the actual output current.

Returns a float with five digits of precision.

#### property current\_setpoint

Control the programmed (set) output current.(dynamic)

## property display

Get the displayed voltage and current.

Returns a list of floating point numbers in the order of [ measured voltage, programmed voltage, measured current, programmed current, over voltage set point, under voltage set point ].

# property foldback\_delay

Control the fold back delay.

Adds an additional delay to the standard fold back delay (250 ms) by multiplying the set value by 0.1. Valid values are integers between 0 to 255.

### property foldback\_enabled

Control the fold back protection of the power supply.

Valid values are True to arm the fold back protection and False to cancel the fold back protection.

## foldback\_reset()

Reset the fold back delay to 0 s, restoring the standard 250 ms delay.

Property is UNTESTED.

# property id

Get the identity of the instrument.

Returns a list of instrument manufacturer and model in the format: ["LAMBDA", "GENX-Y"]

# property last\_test\_date

Get the date of the last test, possibly calibration date.

Returns a string in the format: yyyy/mm/dd.

# property master\_slave\_setting

Get the master and slave settings.

Possible master return values are 1, 2, 3, and 4. The slave value is 0.

Property is UNTESTED.

# property mode

Measure the output mode of the power supply.

When power supply is on, the returned value will be either 'CV' for control voltage or 'CC' for or control current. If the power supply is off, the returned value will be 'OFF'.

# property multidrop\_capability

Get whether the multi-drop option is available on the power supply.

If return value is False, the option is not available, if True it is available.

Property is UNTESTED.

#### property next\_error

Get the next error of the instrument (tuple of code and message).

# property options

Get the device options installed.

## property output\_enabled

Control the output of the power supply.

Valid values are True to turn output on and False to turn output off, shutting down any voltage or current.

# property over\_voltage

Control the over voltage protection. (dynamic)

# property pass\_filter

Control the low pass filter frequency of the A to D converter for voltage and current measurement.

Valid frequency values are 18, 23, or 46 Hz. Default value is 18 Hz.

7.58. TDK Lambda 591

#### ramp\_to\_current(target\_current, steps=20, pause=0.2)

Ramps to a target current from the set current value over a certain number of linear steps, each separated by a pause duration.

#### **Parameters**

- target\_current Target current in amps
- **steps** Integer number of steps
- pause Pause duration in seconds to wait between steps

### read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

### read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- count (int) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

#### **Returns bytes**

Bytes response of the instrument (including termination).

#### recall()

Recall last saved instrument settings.

# property remote

Control the current remote operation of the power supply.

Valid values are 'LOC' for local mode, 'REM' for remote mode, and 'LLO' for local lockout mode.

# property repeat

Measure the last command again.

Returns output of the last command.

# reset()

Reset the instrument to default values.

# save()

Save current instrument settings.

#### property serial

Get the serial number of the instrument.

Returns the serial number of of the instrument as an ASCII string.

# set\_max\_over\_voltage()

Set the over voltage protection to the maximum level for the power supply.

## shutdown()

Safety shutdown the power supply.

Ramps the power supply down to zero current using the self.ramp\_to\_current(0.0) method and turns the output off.

#### property status

Get the power supply status.

Returns a list in the order of [ actual voltage (MV), the programmed voltage (PV), the actual current (MC), the programmed current (PC), the status register (SR), and the fault register (FR) ].

# property under\_voltage

Control the under voltage limit.

Property is UNTESTED. (dynamic)

## property version

Get the software version on instrument.

Returns the software version as an ASCII string.

# property voltage

Measure the actual output voltage.

# property voltage\_setpoint

Control the programmed (set) output voltage.(dynamic)

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

#### **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

## **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

# **Parameters**

- command Command to send.
- values The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

# 7.58.2 TDK Lambda Genesys 80-65 DC power supply

Bases: TDK\_Lambda\_Base

Represents the TDK Lambda Genesys 80-65 DC power supply. Class inherits commands from the TDK\_Lambda\_Base parent class and utilizes dynamic properties adjust valid values on various properties.

7.58. TDK Lambda 593

```
psu = TDK_Gen80_65("COM3", 6)  # COM port and daisy-chain address
psu.remote = "REM"  # PSU in remote mode
psu.output_enabled = True  # Turn on output
psu.ramp_to_current(2.0)  # Ramp to 2.0 A of current
print(psu.current)  # Measure actual PSU current
print(psu.voltage)  # Measure actual PSU voltage
psu.shutdown()  # Run shutdown command
```

The initialization of a TDK instrument requires the current address of the TDK power supply. The default address for the TDK Lambda is 6.

# **Parameters**

- adapter VISAAdapter instance
- name Instrument name. Default is "TDK Lambda Gen80-65"
- address Serial port daisy chain number. Default is 6.

## property address

Set the address of the power supply.

Valid values are integers between 0 - 30 (inclusive).

# property auto\_restart\_enabled

Control the auto restart mode, which restores the power supply to the last output voltage and current settings with output enabled on startup.

Valid values are True to restore output settings with output enabled on startup and False to disable restoration of settings and output disabled on startup.

#### check\_errors()

Read all errors from the instrument and log them.

#### Returns

List of error entries.

# check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Only use this command for setting commands, i.e. non-querying commands.

Any non-querying commands (i.e., a command that does NOT have the "?" symbol in it like the instrument command "PV 10") will automatically return an "OK" reply for valid command or an error code. This is done to confirm that the instrument has received the command. Any querying commands (i.e., a command that does have the "?" symbol in it like the instrument command "PV?") will return the requested value, not the confirmation.

#### clear()

Clear FEVE and SEVE registers to zero.

### property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property current

Measure the actual output current.

Returns a float with five digits of precision.

# property current\_setpoint

Control the programmed (set) output current.(dynamic)

## property display

Get the displayed voltage and current.

Returns a list of floating point numbers in the order of [ measured voltage, programmed voltage, measured current, programmed current, over voltage set point, under voltage set point ].

### property foldback\_delay

Control the fold back delay.

Adds an additional delay to the standard fold back delay (250 ms) by multiplying the set value by 0.1. Valid values are integers between 0 to 255.

# property foldback\_enabled

Control the fold back protection of the power supply.

Valid values are True to arm the fold back protection and False to cancel the fold back protection.

# foldback\_reset()

Reset the fold back delay to 0 s, restoring the standard 250 ms delay.

Property is UNTESTED.

#### property id

Get the identity of the instrument.

Returns a list of instrument manufacturer and model in the format: ["LAMBDA", "GENX-Y"]

# property last\_test\_date

Get the date of the last test, possibly calibration date.

Returns a string in the format: yyyy/mm/dd.

# property master\_slave\_setting

Get the master and slave settings.

Possible master return values are 1, 2, 3, and 4. The slave value is 0.

Property is UNTESTED.

#### property mode

Measure the output mode of the power supply.

When power supply is on, the returned value will be either 'CV' for control voltage or 'CC' for or control current. If the power supply is off, the returned value will be 'OFF'.

7.58. TDK Lambda 595

# property multidrop\_capability

Get whether the multi-drop option is available on the power supply.

If return value is False, the option is not available, if True it is available.

Property is UNTESTED.

#### property next\_error

Get the next error of the instrument (tuple of code and message).

# property options

Get the device options installed.

# property output\_enabled

Control the output of the power supply.

Valid values are True to turn output on and False to turn output off, shutting down any voltage or current.

# property over\_voltage

Control the over voltage protection. (dynamic)

# property pass\_filter

Control the low pass filter frequency of the A to D converter for voltage and current measurement.

Valid frequency values are 18, 23, or 46 Hz. Default value is 18 Hz.

# ramp\_to\_current(target\_current, steps=20, pause=0.2)

Ramps to a target current from the set current value over a certain number of linear steps, each separated by a pause duration.

#### **Parameters**

- target\_current Target current in amps
- **steps** Integer number of steps
- pause Pause duration in seconds to wait between steps

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

Read up to (excluding) *read\_termination* or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

### read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

## **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer.
- **kwargs** Keyword arguments for the adapter.

#### **Returns bytes**

Bytes response of the instrument (including termination).

# recall()

Recall last saved instrument settings.

#### property remote

Control the current remote operation of the power supply.

Valid values are 'LOC' for local mode, 'REM' for remote mode, and 'LLO' for local lockout mode.

#### property repeat

Measure the last command again.

Returns output of the last command.

# reset()

Reset the instrument to default values.

#### save()

Save current instrument settings.

# property serial

Get the serial number of the instrument.

Returns the serial number of of the instrument as an ASCII string.

# set\_max\_over\_voltage()

Set the over voltage protection to the maximum level for the power supply.

#### shutdown()

Safety shutdown the power supply.

Ramps the power supply down to zero current using the self.ramp\_to\_current(0.0) method and turns the output off.

## property status

Get the power supply status.

Returns a list in the order of [ actual voltage (MV), the programmed voltage (PV), the actual current (MC), the programmed current (PC), the status register (SR), and the fault register (FR) ].

# property under\_voltage

Control the under voltage limit.

Property is UNTESTED. (dynamic)

# property version

Get the software version on instrument.

Returns the software version as an ASCII string.

#### property voltage

Measure the actual output voltage.

# property voltage\_setpoint

Control the programmed (set) output voltage.(dynamic)

## wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

# **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

7.58. TDK Lambda 597

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- **\*\*kwargs** (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.59 Tektronix

This section contains specific documentation on the Tektronix instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.59.1 TDS2000 Oscilloscope

Bases: SCPIUnknownMixin, Instrument

Represents the Tektronix TDS 2000 Oscilloscope and provides a high-level for interacting with the instrument

# 7.59.2 AFG3152C Arbitrary function generator

Bases: SCPIUnknownMixin, Instrument

Represents the Tektronix AFG 3000 series (one or two channels) arbitrary function generator and provides a high-level for interacting with the instrument.

```
afg=AFG3152C("GPIB::1")  # AFG on GPIB 1
afg.reset()  # Reset to default
afg.ch1.shape='sinusoidal'  # Sinusoidal shape
afg.ch1.unit='VPP'  # Sets CH1 unit to VPP
afg.ch1.amp_vpp=1  # Sets the CH1 level to 1 VPP
afg.ch1.frequency=1e3  # Sets the CH1 frequency to 1KHz
afg.ch1.enable()  # Enables the output from CH1
```

```
class pymeasure.instruments.tektronix.afg3152c.AFG3152CChannel(parent, id)
     Bases: Channel
     property amp_dbm
           Control the output amplitude in dBm. (float)
     property amp_vpp
           Control the output amplitude in Vpp. (float)
     property amp_vrms
           Control the output amplitude in Vrms. (float)
     property duty
           Control the duty cycle of pulse. (float))
     property frequency
           Control the frequency. (float)
     property impedance
           Control the output impedance of the channel. Be careful with this.
     insert_id(command)
           Prepend the channel id for most writes.
     property offset
           Control the amplitude offset. It is always in Volt. (float)
     property shape
           Control the shape of the output. (str)
     property unit
           Control the amplitude unit. (str)
     waveform(shape='SIN', frequency=1000000.0, units='VPP', amplitude=1, offset=0)
           General setting method for a complete wavefunction
```

# 7.60 Teledyne

This section contains specific documentation on the Teledyne instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

If the instrument you are looking for is not here, also check *LeCroy* for older instruments.

# 7.60.1 Teledyne T3AFG Arbitrary Waveform Generator

```
\textbf{class} \ \ \textbf{pymeasure.instruments.teledyne.TeledyneT3AFG} (a dapter, name = 'Teledyne\ T3AFG', **kwargs')
```

Bases: SCPIMixin, Instrument

Represents the Teledyne T3AFG series of arbitrary waveform generator interface for interacting with the instrument.

Initially targeting T3AFG80, some features may not be available on lower end models and features from higher end models are not included here yet.

Future improvements (help welcomed):

7.60. Teledyne 599

- Add other OUTPut related controls like Load and Polarity
- Add other Basic Waveform related controls like Period
- Add frequency ranges per model
- · Add channel coupling control

#### ch 1

#### Channel

SignalChannel

## $ch_2$

#### Channel

SignalChannel

# check\_errors()

Read all errors from the instrument.

#### Returns

List of error entries.

#### check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

# Returns

List of error entries.

# clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property id

Get the identification of the instrument.

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# property options

Get the device options installed.

# read(\*\*kwargs)

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument.

#### shutdown()

Brings the instrument to a safe and stable state

#### property status

Get the status byte and Master Summary Status bit.

#### wait\_for(query delay=None)

Wait for some time. Used by 'ask' to wait before reading.

# **Parameters**

**query\_delay** – Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

## **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

```
write_binary_values(command, values, *args, **kwargs)
```

Write binary values to the device.

#### **Parameters**

- command Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

# write\_bytes(content, \*\*kwargs)

Write the bytes *content* to the instrument.

class pymeasure.instruments.teledyne.teledyneT3AFG.SignalChannel(parent, id)

Bases: Channel

7.60. Teledyne 601

# property amplitude

Control the amplitude of waveform to be output in volts peak-to-peak. Has no effect when WVTP is NOISE or DC. Max amplitude depends on offset, frequency, and load. Amplitude is also limited by the channel max output amplitude.(dynamic)

# property frequency

Control the frequency of waveform to be output in Hertz. Has no effect when WVTP is NOISE or DC.(dynamic)

# property max\_output\_amplitude

Control the maximum output amplitude of the channel in volts peak to peak.(dynamic)

# property offset

Control the offset of waveform to be output in volts. Has no effect when WVTP is NOISE. Max offset depends on amplitude, frequency, and load. Offset is also limited by the channel max output amplitude.(dynamic)

# property output\_enabled

Control whether the channel output is enabled (boolean).

#### property wavetype

Control the type of waveform to be output. Options are: {'SINE', 'SQUARE', 'RAMP', 'PULSE', 'NOISE', 'ARB', 'DC', 'PRBS', 'IQ'}

There are shared base classes for Teledyne oscilloscopes. These base classes already work directly for some devices, the following are confirmed:

- TeledyneMAUI
  - Teledyne LeCroy HDO6xxx series (e.g. HDO6054B)

If your device is not listed, the base class might already work well enough anyway. If adding a new device, these base classes should limit the amount of new code necessary.

# 7.60.2 Teledyne Oscilloscope base classes

# **Teledyne Oscilloscope**

Bases: SCPIUnknownMixin, Instrument

A base abstract class for any Teledyne Lecroy oscilloscope.

All Teledyne oscilloscopes have a very similar interface, hence this base class to combine them. Note that specific models will likely have conflicts in their interface.

# **Attributes:**

WRITE\_INTERVAL\_S: minimum time between two commands. If a command is received less than WRITE\_INTERVAL\_S after the previous one, the code blocks until at least WRITE\_INTERVAL\_S seconds have passed. Because the oscilloscope takes a non neglibile time to perform some operations, it might be needed for the user to tweak the sleep time between commands. The WRITE\_INTERVAL\_S is set to 10ms as default however its optimal value heavily depends on the actual commands and on the connection type, so it is impossible to give a unique value to fit all cases. An interval between 10ms and 500ms second proved to be good, depending on the commands and connection latency.

#### $ch_1$

#### Channel

TeledyneOscilloscopeChannel

ch 2

#### Channel

TeledyneOscilloscopeChannel

ch\_3

#### Channel

TeledyneOscilloscopeChannel

ch\_4

#### Channel

TeledyneOscilloscopeChannel

#### autoscale()

Autoscale displayed channels.

# property bwlimit

Set the internal low-pass filter for all channels.(dynamic)

# center\_trigger()

Set the trigger levels to center of the trigger source waveform.

#### ch(source)

Get channel object from its index or its name. Or if source is "math", just return the scope object.

# **Parameters**

```
source – can be 1, 2, 3, 4 or C1, C2, C3, C4, MATH
```

#### Returns

handle to the selected source.

# default\_setup()

Set up the oscilloscope for remote operation.

The COMM\_HEADER command controls the way the oscilloscope formats response to queries. This command does not affect the interpretation of messages sent to the oscilloscope. Headers can be sent in their long or short form regardless of the CHDR setting. By setting the COMM\_HEADER to OFF, the instrument is going to reply with minimal information, and this makes the response message much easier to parse. The user should not be fiddling with the COMM\_HEADER during operation, because if the communication header is anything other than OFF, the whole driver breaks down.

# display\_parameter(parameter, channel)

Same as the display\_parameter method in the Channel subclass.

# download\_image()

Get a BMP image of oscilloscope screen in bytearray of specified file format.

# download\_waveform(source, requested\_points=None, sparsing=None)

Get data points from the specified source of the oscilloscope.

The returned objects are two np.ndarray of data and time points and a dict with the waveform preamble, that contains metadata about the waveform.

#### **Parameters**

7.60. Teledyne 603

- **source** measurement source. It can be "C1", "C2", "C3", "C4", "MATH".
- **requested\_points** number of points to acquire. If None the number of points requested in the previous call will be assumed, i.e. the value of the number of points stored in the oscilloscope memory. If 0 the maximum number of points will be returned.
- **sparsing** interval between data points. For example if sparsing = 4, only one point every 4 points is read. If 0 or None the sparsing of the previous call is assumed, i.e. the value of the sparsing stored in the oscilloscope memory.

#### Returns

data\_ndarray, time\_ndarray, waveform\_preamble\_dict: see waveform\_preamble property for dict format.

# property intensity

Set the intensity level of the grid or the trace in percent

### measure\_parameter(parameter, channel)

Same as the measure\_parameter method in the Channel subclass

### property memory\_size

Control the maximum depth of memory (float or string). Assign for example 500, 100e6, "100K", "25MA".

The reply will always be a float.

### run()

Starts repetitive acquisitions.

This is the same as pressing the Run key on the front panel.

# single()

Causes the instrument to acquire a single trigger of data.

This is the same as pressing the Single key on the front panel.

#### stop()

Stops the acquisition. This is the same as pressing the Stop key on the front panel.

### property timebase

Get timebase setup as a dict containing the following keys:

- "timebase\_scale": horizontal scale in seconds/div (float)
- "timebase\_offset": interval in seconds between the trigger and the reference position (float)

#### property timebase\_offset

Control the time interval in seconds between the trigger event and the reference position (at center of screen by default).

# property timebase\_scale

Control the horizontal scale (units per division) in seconds for the main window (float).

# timebase\_setup(scale=None, offset=None)

Set up timebase. Unspecified parameters are not modified. Modifying a single parameter might impact other parameters. Refer to oscilloscope documentation and make multiple consecutive calls to timebase\_setup if needed.

# **Parameters**

• **scale** – interval in seconds between the trigger event and the reference position.

• **offset** – horizontal scale per division in seconds/div.

### property trigger

Get trigger setup as a dict containing the following keys:

- "mode": trigger sweep mode [auto, normal, single, stop]
- "trigger\_type": condition that will trigger the acquisition of waveforms [edge, slew,glit,intv,runt,drop]
- "source": trigger source [c1,c2,c3,c4]
- "hold\_type": hold type (refer to page 172 of programing guide)
- "hold\_value1": hold value1 (refer to page 172 of programing guide)
- "hold\_value2": hold value2 (refer to page 172 of programing guide)
- "coupling": input coupling for the selected trigger sources
- "level": trigger level voltage for the active trigger source
- "level2": trigger lower level voltage for the active trigger source (only slew/runt trigger)
- "slope": trigger slope of the specified trigger source

# property trigger\_mode

Control the trigger sweep mode (string).

<mode>:= {AUTO.NORM.SINGLE.STOP}

- auto: When AUTO sweep mode is selected, the oscilloscope begins to search for the trigger signal that meets the conditions. If the trigger signal is satisfied, the running state on the top left corner of the user interface shows Trig'd, and the interface shows stable waveform. Otherwise, the running state always shows Auto, and the interface shows unstable waveform.
- normal: When NORMAL sweep mode is selected, the oscilloscope enters the wait trigger state and begins to search for trigger signals that meet the conditions. If the trigger signal is satisfied, the running state shows Trig'd, and the interface shows stable waveform. Otherwise, the running state shows Ready, and the interface displays the last triggered waveform (previous trigger) or does not display the waveform (no previous trigger).
- single: When SINGLE sweep mode is selected, the backlight of SINGLE key lights up, the oscilloscope enters the waiting trigger state and begins to search for the trigger signal that meets the conditions. If the trigger signal is satisfied, the running state shows Trig'd, and the interface shows stable waveform. Then, the oscilloscope stops scanning, the RUN/STOP key is red light, and the running status shows Stop. Otherwise, the running state shows Ready, and the interface does not display the waveform.
- stopped: STOP is a part of the option of this command, but not a trigger mode of the oscilloscope.

# property trigger\_select

Control the condition that will trigger the acquisition of waveforms (string).

Depending on the trigger type, additional parameters must be specified. These additional parameters are grouped in pairs. The first in the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs may be given in any order and restricted to those variables to be changed.

There are five parameters that can be specified. Parameters 1. 2. 3. are always mandatory. Parameters 4. 5. are required only for certain combinations of the previous parameters.

- 1. <trig\_type>:={edge, slew, glit, intv, runt, drop}
- 2. <source>:= $\{c1, c2, c3, c4, line\}$

7.60. Teledyne 605

- 3. <hold\_type>:=
  - {ti, off} for edge trigger.
  - {ti} for drop trigger.
  - {ps, pl, p2, p1} for glit/runt trigger.
  - {is, il, i2, i1} for slew/intv trigger.
- 4. <hold value1>:= a time value with unit.
- 5. <hold\_value2>:= a time value with unit.

#### Note:

- "line" can only be selected when the trigger type is "edge".
- All time arguments should be given in multiples of seconds. Use the scientific notation if necessary.
- The range of hold\_values varies from trigger types. [80nS, 1.5S] for "edge" trigger, and [2nS, 4.2S] for others.
- The trigger\_select command is switched automatically between the short, normal and extended version depending on the number of expected parameters.

Set up trigger.

Unspecified parameters are not modified. Modifying a single parameter might impact other parameters. Refer to oscilloscope documentation and make multiple consecutive calls to trigger\_setup and channel\_setup if needed.

#### **Parameters**

- **mode** trigger sweep mode [auto, normal, single, stop]
- **source** trigger source [c1, c2, c3, c4, line]
- **trigger\_type** condition that will trigger the acquisition of waveforms [edge,slew,glit,intv,runt,drop]
- **hold\_type** hold type (refer to page 172 of programing guide)
- **hold\_value1** hold value1 (refer to page 172 of programing guide)
- **hold\_value2** hold value2 (refer to page 172 of programing guide)
- **coupling** input coupling for the selected trigger sources
- **level** trigger level voltage for the active trigger source
- **level2** trigger lower level voltage for the active trigger source (only slew/runt trigger)
- **slope** trigger slope of the specified trigger source

# property waveform\_first\_point

Control the address of the first data point to be sent (int). For waveforms acquired in sequence mode, this refers to the relative address in the given segment. The first data point starts at zero and is strictly positive.

# property waveform\_points

Control the number of waveform points to be transferred with the digitize method (int). NP = 0 sends all data points.

Note that the oscilloscope may provide less than the specified nb of points.

## property waveform\_preamble

Get preamble information for the selected waveform source as a dict with the following keys:

- "requested\_points": number of data points requested by the user (int)
- "sampled points": number of data points sampled by the oscilloscope (int)
- "transmitted points": number of data points actually transmitted (optional) (int)
- "memory\_size": size of the oscilloscope internal memory in bytes (int)
- "sparsing": sparse point. It defines the interval between data points. (int)
- "first\_point": address of the first data point to be sent (int)
- "source": source of the data: "C1", "C2", "C3", "C4", "MATH".
- "grid\_number": number of horizontal grids (it is a read-only property)
- "xdiv": horizontal scale (units per division) in seconds
- "xoffset": time interval in seconds between the trigger event and the reference position
- "ydiv": vertical scale (units per division) in Volts
- "yoffset": value that is represented at center of screen in Volts

## property waveform\_sparsing

Control the interval between data points (integer). For example:

SP = 0 sends all data points. SP = 4 sends 1 point every 4 data points.

# write(command, \*\*kwargs)

Write the command to the instrument through the adapter.

Note: if the last command was sent less than WRITE\_INTERVAL\_S before, this method blocks for the remaining time so that commands are never sent with rate more than 1/WRITE\_INTERVAL\_S Hz.

#### **Parameters**

**command** – command string to be sent to the instrument

# **Teledyne Channel**

 $\textbf{class} \ \ \textbf{pymeasure.instruments.teledyne.teledyne\_oscilloscope.TeledyneOscilloscopeChannel} (\textit{parent}, id)$ 

Bases: Channel

A base abstract class for channel on a *TeledyneOscilloscope* device.

#### property bwlimit

Control the internal low-pass filter for this channel.

The current bandwidths can only be read back for all channels at once! (dynamic)

## property coupling

Control the coupling with a string parameter ("ac 1M", "dc 1M", "ground").

#### property current\_configuration

Get channel configuration as a dict containing the following keys:

- "channel": channel number (int)
- "attenuation": probe attenuation (float)

7.60. Teledyne 607

- "bandwidth\_limit": bandwidth limiting enabled (bool)
- "coupling": "ac 1M", "dc 1M", "ground" coupling (str)
- "offset": vertical offset (float)
- "skew\_factor": channel-tochannel skew factor (float)
- "display": currently displayed (bool)
- "unit": "A" or "V" units (str)
- "volts\_div": vertical divisions (float)
- "inverted": inverted (bool)
- "trigger\_coupling": trigger coupling can be "dc" "ac" "highpass" "lowpass" (str)
- "trigger\_level": trigger level (float)
- "trigger\_level2": trigger lower level for SLEW or RUNT trigger (float)
- "trigger\_slope": trigger slope can be "negative" "positive" "window" (str)

# property display

Control the display enabled state. (strict bool)

# property display\_parameter

Set the waveform processing of this channel with the specified algorithm and the result is displayed on the front panel.

The command accepts the following parameters:

Parameter	Description
PKPK	vertical peak-to-peak
MAX	maximum vertical value
MIN	minimum vertical value
AMPL	vertical amplitude
TOP	waveform top value
BASE	waveform base value
CMEAN	average value in the first cycle
MEAN	average value
RMS	RMS value
CRMS	RMS value in the first cycle
OVSN	overshoot of a falling edge
FPRE	preshoot of a falling edge
OVSP	overshoot of a rising edge
RPRE	preshoot of a rising edge
PER	period
FREQ	frequency
PWID	positive pulse width
NWID	negative pulse width
RISE	rise-time
FALL	fall-time
WID	Burst width
DUTY	positive duty cycle
NDUTY	negative duty cycle
ALL	All measurement

## insert\_id(command)

Insert the channel id in a command replacing *placeholder*.

Subclass this method if you want to do something else, like always prepending the channel id.

#### measure\_parameter(parameter: str)

Process a waveform with the selected algorithm and returns the specified measurement.

#### **Parameters**

parameter - same as the display\_parameter property

## property offset

Control the center of the screen in Volts by a a float parameter. The range of legal values varies depending on range and scale. If the specified value is outside of the legal range, the offset value is automatically set to the nearest legal value.

# property probe\_attenuation

Control the probe attenuation. The probe attenuation may be from 0.1 to 10000.

# property scale

Control the vertical scale (units per division) in Volts.

# setup(\*\*kwargs)

Setup channel. Unspecified settings are not modified.

Modifying values such as probe attenuation will modify offset, range, etc. Refer to oscilloscope documentation and make multiple consecutive calls to setup() if needed. See property descriptions for more information.

#### **Parameters**

- bwlimit
- coupling
- display
- invert
- offset
- skew\_factor
- probe\_attenuation
- scale
- unit
- trigger\_coupling
- trigger\_level
- trigger\_level2
- · trigger\_slope

#### property trigger\_coupling

Control the input coupling for the selected trigger sources (string).

- ac: AC coupling block DC component in the trigger path, removing dc offset voltage from the trigger waveform. Use AC coupling to get a stable edge trigger when your waveform has a large dc offset.
- dc: DC coupling allows dc and ac signals into the trigger path.

7.60. Teledyne 609

- lowpass: HFREJ coupling places a lowpass filter in the trigger path.
- highpass: LFREJ coupling places a highpass filter in the trigger path.

# property trigger\_level

Control the trigger level voltage for the active trigger source (float).

When there are two trigger levels to set, this command is used to set the higher trigger level voltage for the specified source. trigger\_level2 is used to set the lower trigger level voltage.

When setting the trigger level it must be divided by the probe attenuation. This is not documented in the datasheet and it is probably a bug of the scope firmware. An out-of-range value will be adjusted to the closest legal value.

# property trigger\_slope

Control the trigger slope of the specified trigger source (string).

<trig\_slope>:={NEG,POS,WINDOW} for edge trigger <trig\_slope>:={NEG,POS} for other trigger

parameter	trigger slope
negative	Negative slope for edge trigger or other trigger
positive	Positive slope for edge trigger or other trigger
window	Window slope for edge trigger

(dynamic)

# **Teledyne MAUI Oscilloscope**

class pymeasure.instruments.teledyne.TeledyneMAUI(adapter, name='Teledyne Oscilloscope', \*\*kwargs)

Bases: TeledyneOscilloscope

A base class for the MAUI-type of Teledyne oscilloscopes.

This base class works out of the box. Some properties, especially the number of channels, might have to be adjusted to the actual device.

The manual detailing the API is "MAUI Oscilloscopes Remote Control and Automation Manual" (link).

This class of Teledyne oscilloscopes also support direct VBS commands. See vbs\_ask() and vbs\_write().

#### $ch_1$

## Channel

**TeledyneMAUIChannel** 

ch\_2

#### Channel

TeledyneMAUIChannel

ch\_3

## Channel

*TeledyneMAUIChannel* 

ch\_4

#### Channel

TeledyneMAUIChannel

# download\_image(\*\*kwargs)

Get a BMP image of oscilloscope screen in bytearray of specified file format.

The hardcopy destination is set to "REMOTE" by default.

#### **Parameters**

\*\*kwargs – Keyword arguments for hardcopy\_setup()

# force\_trigger()

Make one acquisition if in active trigger mode.

No action is taken if the device is in 'Stop trigger mode'.

# hardcopy\_setup(\*\*kwargs)

Specify hardcopy settings.

Connect a printer or define how to save to file. Set any or all of the following parameters.

#### **Parameters**

- device {BMP, JPEG, PNG, TIFF}
- **format** {PORTRAIT, LANDSCAPE}
- background {Std, Print, BW}
- **destination** {PRINTER, CLIPBOARD, EMAIL, FILE, REMOTE}
- area {GRIDAREAONLY, DSOWINDOW, FULLSCREEN}
- directory Any legal DOS path, for FILE mode only
- **filename** Filename string, no extension, for FILE mode only
- printername Valid printer name, for PRINTER mode only
- $portname \{GPIB, NET\}$

# property hardcopy\_setup\_current

Get current hardcopy config.

# property trigger

Get trigger setup as a dict containing the following keys:

- "mode": trigger sweep mode [auto, normal, single, stop]
- "trigger type": condition that will trigger the acquisition of waveforms [edge,slew,glit,intv,runt,drop]
- "source": trigger source [c1,c2,c3,c4]
- "hold\_type": hold type (refer to page 172 of programing guide)
- "hold\_value1": hold value1 (refer to page 172 of programing guide)
- "hold\_value2": hold value2 (refer to page 172 of programing guide)
- "coupling": input coupling for the selected trigger sources
- "level": trigger level voltage for the active trigger source
- "slope": trigger slope of the specified trigger source

7.60. Teledyne 611

```
vbs_ask(name: str) \rightarrow str
```

Return the value of a VBS variable.

Only the target needs to be specified, a query is formatted by this method. Note: the target name is not escaped!

See *vbs\_write()* for more info.

A very basic example of usage:

```
instrument.vbs_ask("app.Display.GridMode")
```

## vbs\_write(message: str)

Write a VBS command directly to the device.

This class of oscilloscopes also allows the direct usage of Visual Basic Scripting (VBScript). With this method a literal VBS command is sent. You can use the 'MAUI Browser' on the oscilloscope to list all available variables.

A very basic example of usage:

```
instrument.vbs_write("app.Display.GridMode = Dual")
```

# **Teledyne MAUI Channel**

class pymeasure.instruments.teledyne.teledyneMAUI.TeledyneMAUIChannel(parent, id)

Bases: TeledyneOscilloscopeChannel

Base class for channels on a TeledyneMAUI device.

## autoscale()

Perform auto-setup command for channel.

# property current\_configuration

Get channel configuration as a dict containing the following keys:

- "channel": channel number (int)
- "attenuation": probe attenuation (float)
- "bandwidth\_limit": bandwidth limiting, parsed for this channel (str)
- "coupling": "ac 1M", "dc 1M", "ground" coupling (str)
- "offset": vertical offset (float)
- "display": currently displayed (bool)
- "volts\_div": vertical divisions (float)
- "trigger\_coupling": trigger coupling can be "dc" "ac" "highpass" "lowpass" (str)
- "trigger\_level": trigger level (float)
- "trigger\_slope": trigger slope can be "negative" "positive" "window" (str)

# setup(\*\*kwargs)

Setup channel. Unspecified settings are not modified.

Modifying values such as probe attenuation will modify offset, range, etc. Refer to oscilloscope documentation and make multiple consecutive calls to setup() if needed. See property descriptions for more information.

#### **Parameters**

- bwlimit
- · coupling
- display
- offset
- probe\_attenuation
- scale
- trigger\_coupling
- trigger\_level
- trigger\_slope

# 7.61 Temptronic

This section contains specific documentation on the temptronic instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.61.1 Temptronic Base Class

Relevant flags are:

```
class pymeasure.instruments.temptronic.ATSBase(adapter, name='ATSBase', **kwargs)
    Bases: SCPIUnknownMixin, Instrument
    The base class for Temptronic ATSXXX instruments.

property air_temperature
    Get air temperature in 0.1 °C increments.

    Type
        float

at_temperature()

    Returns
        True if at temperature.

property auxiliary_condition_code

Get out auxiliary condition status register.

Type
    int
```

7.61. Temptronic 613

Bit	Meaning
10	None
9	Ramp mode
8	Mode: 0 programming, 1 manual
7	None
6	TS status: 0 start-up, 1 ready
5	Flow: 0 off, 1 on
4	Sense mode: 0 air, 1 DUT
3	Compressor: 0 on, 1 off (heating possible)
2	Head: 0 lower, upper
1	None
0	None

Refer to chapter 4 in the manual

# clear()

Clear device-specific errors.

See *error\_code* for further information.

# property compressor\_enable

Control whether the compressor is enabled: True enables compressors, False disables it.

#### **Type**

Boolean

**configure**(temp\_window=1, dut\_type='T', soak\_time=30, dut\_constant=100, temp\_limit\_air\_low=-60, temp\_limit\_air\_high=220, temp\_limit\_air\_dut=50, maximum\_test\_time=1000)

Convenience method for most relevant configuration properties.

#### **Parameters**

- dut\_type string: indicating which DUT type to use
- soak\_time float: elapsed time in soak\_window before settling is indicated
- soak\_window float: Soak window size or temperature settlings bounds (K)
- **dut\_constant** float: time constant of DUT, higher values indicate higher thermal mass
- temp\_limit\_air\_low float: minimum flow temperature limit (°C)
- **temp\_limit\_air\_high** float: maximum flow temperature limit (°C)
- **temp\_limit\_air\_dut** float: allowed temperature difference (K) between DUT and Flow
- maximum\_test\_time float: maximum test time (seconds) for a single temperature point (safety)

#### **Returns**

self

# property copy\_active\_setup\_file

Set active setup file (0) to copy to setup n(1 - 12).

# Type

int

# property current\_cycle\_count

Get the number of cycles to do

Type

int

# property cycling\_enable

Control whether cycling is enabled.

**Type** 

bool

cycling\_enable = True (start cycling) cycling\_enable = False (stop cycling)

# cycling\_stopped()

#### **Returns**

True if cycling has stopped.

# property dut\_constant

Control thermal constant (default 100) of DUT.

**Type** 

float

Lower values indicate lower thermal mass, higher values indicate higher thermal mass respectively.

# property dut\_mode

Control DUT mode ("On" or "OFF")

**Type** 

string

# property dut\_temperature

Get DUT temperature, in 0.1 °C increments.

**Type** 

float

# property dut\_type

Control DUT sensor type.

Type

string

Possible values are:

String	Meaning
٠,	no DUT
'T'	T-DUT
'K'	K-DUT

Warning: If in DUT mode without DUT being connected, TS flags DUT error

# property dynamic\_temperature\_setpoint

Get the dynamic temperature setpoint.

**Type** 

float

7.61. Temptronic 615

```
property enable_air_flow
     Set TS air flow.
     True enables air flow, False disables it
          Type
               bool
end_of_all_cycles()
          Returns
               True if cycling has stopped.
end_of_one_cycle()
          Returns
               True if TS is at end of one cycle.
end_of_test()
          Returns
               True if TS is at end of test.
enter_cycle()
     Enter Cycle by sending RMPC 1.
          Returns
               self
enter_ramp()
     Enter Ramp by sending RMPS 0.
          Returns
               self
property error_code
     Get the device-specific error register (16 bits).
          Type
               ErrorCode
error_status()
     Returns error status code (maybe used for logging).
          Returns
               ErrorCode
property head
     Control TS head position.
          Type
               string
      down: transfer head to lower position up: transfer head to elevated position
property learn_mode
     Control DUT automatic tuning (learning).
          Type
               bool False: off True: automatic tuning on
```

property load\_setup\_file

```
Set which setup file to load and load it.
      Valid range is between 1 to 12.
           Type
               int
property local_lockout
     Control whether to lock locally: True disables TS GUI, False enables it.
property main_air_flow_rate
     Get main nozzle air flow rate in liters/sec.
property maximum_test_time
      Control maximum allowed test time (s).
           Type
               float
     This prevents TS from staying at a single temperature forever. Valid range: 0 to 9999
property mode
      Get a string indicating what the system is doing at the time the query is processed.
           Type
               string
      (dynamic)
next_setpoint()
      Step to the next setpoint during temperature cycling.
not_at_temperature()
           Returns
               True if not at temperature.
property nozzle_air_flow_rate
      Get main nozzle air flow rate in scfm.
property ramp_rate
     Control ramp rate (K / min).
           Type
               float
     allowed values: nn.n: 0 to 99.9 in 0.1 K per minute steps. nnnn: 100 to 9999 in 1 K per minute steps.
property remote_mode
     Control whether it is in remote mode.
reset()
     Reset (force) the System to the Operator screen.
           Returns
               self
property set_point_number
```

7.61. Temptronic 617

Control the setpoint index to be the current setpoint.

```
Type
               int
     Valid range is 0 to 17 when on the Cycle screen or or 0 to 2 in case of operator screen (0=hot, 1=ambient,
     2=cold).
set_temperature(set_temp)
     sweep to a specified setpoint.
          Parameters
               set_temp – target temperature for DUT (float)
          Returns
               self
shutdown(head=False)
     Turn down TS (flow and remote operation).
          Parameters
              head - Lift head if True
          Returns
               self
start(enable_air_flow=True)
     start TS in remote mode.
          Parameters
               enable_air_flow - flow starts if True
          Returns
               self
property temperature
     Get current temperature with 0.1 °C resolution.
          Type
               float
     Temperature readings origin depends on dut_mode setting. Reading higher than 400 (°C) indicates inva-
     lidity.
property temperature_condition_status_code
     Get temperature condition status register.
          Type
               TemperatureStatusCode
property temperature_event_status
     Get temperature event status register.
          Type
               TemperatureStatusCode
     Hint: Reading will clear register content.
property temperature_limit_air_dut
     Get air to DUT temperature limit.
```

**Type** 

float

Allowed difference between nozzle air and DUT temperature during settling. Valid range between 10 to 300 °C in 1 degree increments.

# property temperature\_limit\_air\_high

Control upper air temperature limit.

```
Type
```

float

Valid range between 25 to 255 (°C). Setpoints above current value cause "out of range" error in TS.

## property temperature\_limit\_air\_low

Control lower air temperature limit.

## Type

float

Valid range between -99 to 25 (°C). Setpoints below current value cause "out of range" error in TS. (dynamic)

## property temperature\_setpoint

Control selected setpoint's temperature.

# Type

float

Valid range is -99.9 to 225.0 (°C) or as indicated by temperature\_limit\_air\_high and temperature\_limit\_air\_low. Use convenience function set\_temperature() to prevent unexpected behavior.

# property temperature\_setpoint\_window

Control setpoint's temperature window.

## **Type**

float

Valid range is between 0.1 to 9.9 (°C). Temperature status register flags at temperature in case soak time elapsed while temperature stays in between bounds given by this value around the current setpoint.

#### property temperature\_soak\_time

Control the soak time for the currently selected setpoint.

# **Type**

float

Valid range is between 0 to 9999 (s). Lower values shorten cycle times. Higher values increase cycle times, but may reduce settling errors. See *temperature\_setpoint\_window* for further information.

# property total\_cycle\_count

Control current cycle count (1 - 9999).

## **Type**

int

Sending 0 will stop cycling

# wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

# **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

7.61. Temptronic 619

wait\_for\_settling(time\_limit=300)

block script execution until TS is settled.

# **Parameters**

**time\_limit** – set the maximum blocking time within TS has to settle (float).

Returns

self

Script execution is blocked until either TS has settled or time\_limit has been exceeded (float).

class pymeasure.instruments.temptronic\_temptronic\_base.TemperatureStatusCode(value,

names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Temperature status enums based on IntFlag

Used in conjunction with temperature\_condition\_status\_code.

Value	Enum
32	CYCLING_STOPPED
16	END_OF_ALL_CYCLES
8	END_OF_ONE_CYCLE
4	END_OF_TEST
2	NOT_AT_TEMPERATURE
1	AT_TEMPERATURE
0	NO_STATUS

Error code enums based on IntFlag.

Used in conjunction with error\_code.

Value	Enum
16384	NO_DUT_SENSOR_SELECTED
4096	BVRAM_FAULT
2048	NVRAM_FAULT
1024	NO_LINE_SENSE
512	FLOW_SENSOR_HARDWARE_ERROR
128	INTERNAL_ERROR
32	AIR_SENSOR_OPEN
16	LOW_INPUT_AIR_PRESSURE
8	LOW_FLOW
2	AIR_OPEN_LOOP
1	OVERHEAT
0	OK

# 7.61.2 Temptronic ATS525 Thermostream

Bases: ATSBase

Represent the TemptronicATS525 instruments.

# property system\_current

Get operating current.

# 7.61.3 Temptronic ATS545 Thermostream

Bases: ATSBase

Represents the TemptronicATS545 instrument.

Coding example

```
ts = ATS545('ASRL3::INSTR') # replace adapter address
ts.configure() # basic configuration (defaults to T-DUT)
ts.start() # starts flow (head position not changed)
ts.set_temperature(25) # sets temperature to 25 degC
ts.wait_for_settling() # blocks script execution and polls for settling
ts.shutdown(head=False) # disables thermostream, keeps head down
```

# next\_setpoint()

not implemented in ATS545

set self.set\_point\_number instead

7.61. Temptronic 621

# 7.61.4 Temptronic ECO560 Thermostream

Bases: ATSBase

Represent the TemptronicECO560 instruments.

# **7.62 TEXIO**

This section contains specific documentation on the TEXIO instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.62.1 TEXIO PSW-360L30 Power Supply

Bases: Keithley2260B

Represents the TEXIO PSW-360L30 Power Supply (minimal implementation) and provides a high-level interface for interacting with the instrument.

For a connection through tcpip, the device only accepts connections at port 2268, which cannot be configured otherwise. example connection string: 'TCPIP::xxx.xxx.xxx.xxx::2268::SOCKET'

For a connection through USB on Linux, the kernel is going to create a /dev/ttyACMX device automatically. The serial connection properties are fixed at 9600–8-N-1.

The read termination for this interface is Line-Feed n.

This driver inherits from the Keithley 2260B one. All instructions implemented in the Keithley 2260B driver are also available for the TEXIO PSW-360L30 power supply.

The only addition is the "output" property that is just an alias for the "enabled" property of the Keithley 2260B. Calling the output switch "enabled" is confusing because it is not clear if the whole device is enabled/disable or only the output.

```
source = TexioPSW360L30("TCPIP::xxx.xxx.xxx.xxx::2268::SOCKET")
source.voltage = 1
print(source.voltage)
print(source.current)
print(source.power)
print(source.applied)
```

# property applied

Control voltage (volts) and current (amps) simultaneously. Values need to be supplied as tuple of (voltage, current). Depending on whether the instrument is in constant current or constant voltage mode, the values achieved by the instrument will differ from the ones set.

#### check\_errors()

Log any system errors reported by the instrument.

## check\_get\_errors()

Check for errors after having gotten a property and log them.

Called if check\_get\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# clear()

Clear the instrument status byte.

# property complete

Get the synchronization bit.

This property allows synchronization between a controller and a device. The Operation Complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

# property current

Get the current (in Ampere) the dc power supply is putting out.

# property current\_limit

Control the source current in amps. This is not checked against the allowed range. Depending on whether the instrument is in constant current or constant voltage mode, this might differ from the actual current achieved. (float)

# property enabled

Control whether the output is enabled, see *output\_enabled*.

# property error

Get the next error of the instrument (list of code and message).

#### property id

Get the identification of the instrument.

#### property next\_error

Get the next error in the queue. If you want to read and log all errors, use check\_errors() instead.

# property options

Get the device options installed.

#### property output\_enabled

Control whether the source is enabled, takes values True or False. (bool)

## property power

Get the power (in Watt) the dc power supply is putting out.

7.62. TEXIO 623

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

Read up to (excluding) read\_termination or the whole read buffer.

# read\_binary\_values(\*\*kwargs)

Read binary values from the device.

# read\_bytes(count, \*\*kwargs)

Read a certain number of bytes from the instrument.

#### **Parameters**

- **count** (*int*) Number of bytes to read. A value of -1 indicates to read the whole read buffer
- **kwargs** Keyword arguments for the adapter.

# **Returns bytes**

Bytes response of the instrument (including termination).

#### reset()

Reset the instrument.

#### shutdown()

Disable output, call parent function

#### property status

Get the status byte and Master Summary Status bit.

## property voltage

Get the voltage (in Volt) the dc power supply is putting out.

# property voltage\_setpoint

Control the source voltage in volts. This is not checked against the allowed range. Depending on whether the instrument is in constant current or constant voltage mode, this might differ from the actual voltage achieved. (float)

#### wait\_for(query\_delay=None)

Wait for some time. Used by 'ask' to wait before reading.

## **Parameters**

query\_delay - Delay between writing and reading in seconds. None is default delay.

```
write(command, **kwargs)
```

Write a string command to the instrument appending write\_termination.

#### **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# write\_binary\_values(command, values, \*args, \*\*kwargs)

Write binary values to the device.

# **Parameters**

- **command** Command to send.
- **values** The values to transmit.
- \*\*kwargs (\*args,) Further arguments to hand to the Adapter.

```
write_bytes(content, **kwargs)
```

Write the bytes *content* to the instrument.

# 7.63 Thermotron

This section contains specific documentation on the Thermotron instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.63.1 Thermotron 3800 Oven

Bases: Instrument

Represents the Thermotron 3800 Oven. For now, this driver only supports using Control Channel 1. There is a 1000ms built in wait time after all write commands.

**class Thermotron3800Mode**(value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None)

Bases: IntFlag

Bit	Mode
0	Program mode
1	Edit mode (controller in stop mode)
2	View program mode
3	Edit mode (controller in hold mode)
4	Manual mode
5	Delayed start mode
6	Unused
7	Calibration mode

# property id

Get the instrument identification

## **Returns**

String

# initalize\_oven(wait=True)

The manufacturer recommends a 3 second wait time after after initializing the oven. The optional "wait" variable should remain true, unless the 3 second wait time is taken care of on the user end. The wait time is split up in the following way: 1 second (built into the write function) + 2 seconds (optional wait time from this function (initialize\_oven)).

#### Returns

None

# property mode

Get the operating mode of the oven.

#### Returns

Tuple(String, int)

7.63. Thermotron 625

#### run()

Starts temperature forcing. The oven will ramp to the setpoint.

# Returns

None

# property setpoint

Control the setpoint of the oven in Celsius. (float) "setpoint" will not update until the "run()" command is called. After setpoint is set to a new value, the "run()" command must be called to tell the oven to run to the new temperature.

### Returns

None

# stop()

Stops temperature forcing on the oven.

#### Returns

None

# property temperature

Get the current temperature of the oven via built in thermocouple. Default unit is Celsius, unless changed by the user.

#### Returns

float

#### write(command)

Write a string command to the instrument appending write\_termination.

## **Parameters**

- **command** command string to be sent to the instrument
- **kwargs** Keyword arguments for the adapter.

# 7.64 Thorlabs

This section contains specific documentation on the Thorlabs instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.64.1 Thorlabs PM100USB Powermeter

Bases: SCPIUnknownMixin, Instrument

Represents Thorlabs PM100USB powermeter.

# property energy

Measure the energy in J.

# property power

Measure the power in W.

#### property wavelength

Control the wavelength in nm.

# property wavelength\_max

Measure maximum wavelength, in nm

## property wavelength\_min

Measure minimum wavelength, in nm

# 7.64.2 Thorlabs Pro 8000 modular laser driver

Bases: SCPIUnknownMixin, Instrument

Represents Thorlabs Pro 8000 modular laser driver

#### property LDCCurrent

Control laser current.

# property LDCCurrentLimit

Set Software current Limit (value must be lower than hardware current limit).

# property LDCPolarity

Set laser diode polarity. Allowed values are: ['AG', 'CG']

## property LDCStatus

Set laser diode status. Allowed values are: ['ON', 'OFF']

## property TEDSetTemperature

Control TEC temperature

# property TEDStatus

Control TEC status. Allowed values are: ['ON', 'OFF']

#### property slot

Control slot selection. Allowed values are: range(1, 9)

# 7.65 Thyracont

This section contains specific documentation on the Thyracont instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.65.1 Smartline V1 Transmitter Series

Bases: Instrument

Thyracont Vacuum Instruments Smartline gauges with Communication Protocol V1.

Devices using Protocol V1 were manufactured until 2017.

7.65. Thyracont 627

Connection to the device is made through an RS485 serial connection. The default communication settings are baudrate 9600, 8 data bits, 1 stop bit, no parity, no handshake.

A communication packages is structured as follows:

Characters 0-2: Address for communication Character 3: Command character, uppercase letter for reading and lowercase for writing Characters 4-n: Data for the command, can be empty. Character n+1: Checksum calculated by: (sum of the decimal value of bytes 0-n) mod 64 + 64 Character n+2: Carriage return

#### **Parameters**

- adapter pyvisa resource name of the instrument or adapter instance
- name (string) Name of the instrument.
- address (int) RS485 adddress of the instrument 1-15.
- baud\_rate (int) baudrate used for the communication with the device.
- **kwargs** Any valid key-word argument for Instrument

# property cathode\_enabled

Control the hot/cold cathode state of the pressure gauge.

## check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

If you override this method, you may choose to raise an Exception for certain errors.

#### Returns

List of error entries.

# property device\_type

Get the device type.

# property display\_unit

Control the display's pressure unit.

# property pressure

Get the pressure measurement in mbar.

# read()

Reads a response message from the instrument.

This method also checks for a correct checksum.

#### **Returns**

the data fields

# **Return type**

string

#### **Raises**

**ValueError** – if a checksum error is detected

#### write(command)

Writes a command to the instrument.

This method adds the required address and checksum.

#### **Parameters**

**command** (str) – command to be sent to the instrument

# 7.65.2 Smartline V2 Transmitter Series

Bases: Instrument

A Thyracont vacuum sensor transmitter of the Smartline V2 series.

You may subclass this Instrument and add the appropriate channels, see the following example.

```
from pymeasure.instruments import Instrument
from pymeasure.instruments.thyractont import SmartlineV2

PiezoAndPiraniInstrument(SmartlineV2):
   piezo = Instrument.ChannelCreator(Piezo)
   pirani = Instrument.ChannelCreator(Pirani)
```

#### Communication Protocol v2 via RS485:

- Everything is sent as ASCII characters
- Package (bytes and usage):
  - 0-2 address, 3 access code, 4-5 command, 6-7 data length.
  - if data: 8-n data to be sent, n+1 checksum, n+2 carriage return
  - if no data: 8 checksum, 9 carriage return
- Access codes (request: master->transmitter, response: transmitter->master):
  - read: 0, 1
  - write: 2, 3
  - factory default: 4,5
  - error: -, 7
  - binary 8, 9
- Data length is number of data in bytes (padding with zeroes on left)
- Checksum: Add the decimal numbers of the characters before, mod 64, add 64, show as ASCII.

## **Parameters**

**adress** – The device address in the range 1-16.

**class Sources**(*value*, *names*=<*not given*>, \**values*, *module*=*None*, *qualname*=*None*, *type*=*None*, *start*=1, *boundary*=*None*)

Bases: IntEnum

# property analog\_output\_setting

Get current analog output setting. See manual.

ask(command\_message, query\_delay=None)

Ask for some value and check that the response matches the original command.

7.65. Thyracont 629

#### **Parameters**

 $command\_message (str)$  – Access code, command, length, and content. The command sent is compared to the response command.

# ask\_manually(accessCode, command, data=", query\_delay=None)

Send a message to the transmitter and return its answer.

#### **Parameters**

- accessCode How to access the device.
- **command** Command to send to the device.
- data Data for the command.
- query\_delay (float) Time to wait between writing and reading in seconds.

# **Return str**

Response from the device after error checking.

# property baud\_rate

Set the device baud rate.

# property bootloader\_version

Get the bootloader version.

# check\_set\_errors()

Check the errors after setting a property.

## property device\_address

Set the device address.

# property device\_serial

Get the transmitter device serial number.

# property device\_type

Get the device type, like 'VSR205'.

# property device\_version

Get the device hardware version.

# property display\_data

Control the display data source (strict SOURCES).

#### property display\_orientation

Control the orientation of the display in relation to the pipe ('top', 'bottom').

## property display\_unit

Control the unit shown in the display. ('mbar', 'Torr', 'hPa')

## property firmware\_version

Get the firmware version.

#### get\_sensor\_transition()

Get the current sensor transition between sensors.

## return interpretation:

direct

switch at 1 mbar.

#### • continuous

switch between 5 and 15 mbar.

#### • F[float]T[float]

switch between low and high value.

# • D[float]

switch at value.

## property operating\_hours

Measure the operating hours.

#### property pressure

Get the current pressure of the default sensor in mbar

### property product\_name

Get the product name (article number).

## property range

Get the measurement range in mbar.

## read(command=None)

Read from the device and do error checking.

#### **Parameters**

**command** (*str*) – Original command sent to the device to compare it with the response. None deactivates the check.

# property sensor\_serial

Get the sensor head serial number.

# set\_continuous\_sensor\_transition(low, high)

Set the sensor transition mode to "continuous" mode between *low* and *high* (floats).

# set\_default\_sensor\_transition()

Set the senstor transition mode to the default value, depends on the device.

# set\_direct\_sensor\_transition(transition\_point)

Set the sensor transition to "direct" mode.

## **Parameters**

**transition\_point** (*float*) – Switch between the sensors at that value.

# set\_high(high=")

Set the high pressure to *high* pressure in mbar.

## set\_low(low=")

Set the low pressure to *low* pressure in mbar.

## write(command)

Write a command to the device.

# write\_composition(accessCode, command, data=")

Write a command with an accessCode and optional data to the device.

## **Parameters**

- accessCode How to access the device.
- **command** Two char command string to send to the device.

7.65. Thyracont 631

```
• data – Data for the command.
\textbf{class} \ \ \textbf{pymeasure.instruments.thyracont.smartline\_v2.VSH} (\textit{adapter}, \textit{name} = \textit{Thyracont SmartlineV2})
                                                                  Transmitter', baud rate=115200,
                                                                  address=1, timeout=250, **kwargs)
     Bases: SmartlineV2
     Vacuum transmitter of VSH series with both a pirani and a hot cathode sensor.
     hotcathode
                Channel
                    HotCathode
     pirani
                Channel
                    Pirani
class pymeasure.instruments.thyracont.smartline_v2.VSM(adapter, name="Thyracont SmartlineV2")
                                                                  Transmitter', baud_rate=115200,
                                                                  address=1, timeout=250, **kwargs)
     Bases: SmartlineV2
     Vacuum transmitter of VSM series with both piece and cold cathode sensor.
     coldcathode
                Channel
                    ColdCathode
     piezo
                Channel
                    Piezo
class pymeasure.instruments.thyracont.smartline_v2.VSP(adapter, name="Thyracont SmartlineV2")
                                                                  Transmitter', baud rate=115200,
                                                                  address=1, timeout=250, **kwargs)
     Bases: SmartlineV2
     Vacuum transmitter of VSP/VCP series with a piezo sensor.
     piezo
                Channel
                    Piezo
class pymeasure.instruments.thyracont.smartline_v2.VSR(adapter, name="Thyracont SmartlineV2")
                                                                  Transmitter', baud_rate=115200,
                                                                  address=1, timeout=250, **kwargs)
     Bases: SmartlineV2
     Vacuum transmitter of VSR/VCR series with both a piezo and a pirani sensor.
```

Channel Piezo

piezo

pirani

# Channel

Pirani

# 7.66 Toptica

This section contains specific documentation on the Toptica Photonics instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.66.1 Toptica IBeam Smart Laser diode

Bases: Instrument

IBeam Smart laser diode

For the usage of the different diode driver channels, see the manual

```
laser = IBeamSmart("SomeResourceString")
laser.emission = True
laser.ch_2.power = 1000 # μW
laser.ch_2.enabled = True
laser.shutdown()
```

# **Parameters**

- adapter pyvisa resource name or adapter instance.
- baud\_rate The baud rate you have set in the instrument.
- \*\*kwargs Any valid key-word argument for VISAAdapter.

ch\_1

# Channel

DriverChannel

ch\_2

#### Channel

DriverChannel

ch\_3

## Channel

DriverChannel

ch\_4

# Channel

DriverChannel

7.66. Toptica 633

#### $ch_5$

#### Channel

**DriverChannel** 

## property channel1\_enabled

Control status of Channel 1 of the laser (bool).

Deprecated since version 0.12: Use ch\_1.enabled instead.

# property channel2\_enabled

Control status of Channel 2 of the laser (bool).

Deprecated since version 0.12: Use ch\_2.enabled instead.

# check\_set\_errors()

Check for errors after having gotten a property and log them.

Checks if the last reply is only '[OK]', otherwise a ValueError is raised and the read buffer is flushed because one has to assume that some communication is out of sync.

## property current

Measure the laser diode current in mA.

# disable()

Shutdown all laser operation.

#### property emission

Control emission status of the laser diode driver (bool).

## enable\_continous()

Enable countinous emmission mode.

# enable\_pulsing()

Enable pulsing mode.

The optical output is controlled by a digital input signal on a dedicated connnector on the device.

## property laser\_enabled

Control emission status of the laser diode driver (bool).

Deprecated since version 0.12: Use attr:emission instead.

# property power

Control actual output power in  $\mu W$  of the laser system. In pulse mode this means that the set value might not correspond to the readback one (float up to 200000).

## read()

Read a reply of the instrument and extract the values, if possible.

Reads a reply of the instrument which consists of at least two lines. The initial ones are the reply to the command while the last one should be '[OK]' which acknowledges that the device is ready to receive more commands.

Note: '[OK]' is always returned as last message even in case of an invalid command, where a message indicating the error is returned before the '[OK]'

Value extraction: extract <value> from 'name = <value> [unit]'. If <value> can not be identified the original string is returned.

# Returns

string containing the ASCII response of the instrument (without '[OK]').

#### property serial

Get Serial number of the laser system.

#### shutdown()

Brings the instrument to a safe and stable state.

## property system\_temp

Measure base plate (heatsink) temperature in degree centigrade.

#### property temp

Measure the temperature of the laser diode in degree centigrade.

# property version

Get Firmware version number.

class pymeasure.instruments.toptica.ibeamsmart.DriverChannel(parent, id)

Bases: Channel

A laser diode driver channel for the IBeam Smart laser.

# property enabled

Control the enabled state of the driver channel.

# property power

Set the output power in  $\mu W$  (float up to 200000).

# 7.67 Velleman

This section contains specific documentation on the Velleman instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.67.1 Velleman K8090 8-channel relay board

Bases: Instrument

For usage with the K8090 relay board, by Velleman.

View the "K8090/VM8090 PROTOCOL MANUAL" for the serial command instructions.

The communication is done by serial USB. The IO settings are fixed:

Baud rate	19200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

A short timeout is recommended, since the device is not consistent in giving status messages and serial timeouts will occur also in normal operation.

Use the class like:

7.67. Velleman 635

```
from pymeasure.instruments.velleman import VellemanK8090, VellemanK8090Switches as_
    Switches

instrument = VellemanK8090("ASRL1::INSTR")

# Get status update from device
last_on, curr_on, time_on = instrument.status

# Toggle a selection of channels on
instrument.switch_on = Switches.CH3 | Switches.CH4 | Switches.CH5
```

#### check\_set\_errors()

Check for errors after having set a property and log them.

Called if check\_set\_errors=True is set for that property.

The K8090 replies with a status after a switch command, but **only** after any switch actually changed. In order to guarantee the buffer is empty, we attempt to read it fully here. No actual error checking is done here!

#### Returns

List of error entries.

# read(\*\*kwargs)

The read command specifically for the protocol of the K8090.

This overrides the method from the instrument class.

See write(), replies from the machine use the same format.

A read will return a list of CMD, MASK, PARAM1 and PARAM2.

#### property status

Get current relay status. The reply has a different command byte than the request.

Three items (VellemanK8090Switches flags) are returned:

- Previous state: the state of each relay before this event
- Current state: the state of each relay now
- Timer state: the state of each relay timer

# property switch\_off

Set channels to off state. See switch\_on for more details.

#### property switch\_on

Set channels to on state. Other channels are unaffected. Pass either a list or set of channel numbers (starting at 1), or pass a bitmask.

After switching this waits for a reply from the device. This is only send when a relay actually toggles, otherwise expect a blocking time equal to the communication timeout If speed is important, avoid calling *switch*\_ unnecessarily.

## property version

```
Get firmware version, as (year - 2000, week). E.g. (10, 1)
```

## write(command, \*\*kwargs)

The write command specifically for the protocol of the K8090.

This overrides the method from the Instrument class.

Each packet to the device is 7 bytes:

```
STX (0x04) - CMD - MASK - PARAM1 - PARAM2 - CHK - ETX (0x0F)
```

Where CHK is checksum of the package.

#### **Parameters**

```
{f command}\ (str) – String like "CMD[, MASK, PARAM1, PARAM2]" - only CMD is mandatory
```

Bases: IntFlag

Use to identify switch channels.

# 7.68 Yokogawa

This section contains specific documentation on the Yokogawa instruments that are implemented. If you are interested in an instrument not included, please consider *adding the instrument*.

# 7.68.1 Yokogawa 7651 Programmable Supply

Bases: SCPIUnknownMixin, Instrument

Represents the Yokogawa 7651 Programmable DC Source and provides a high-level for interacting with the instrument.

```
apply_current(max_current=0.001, compliance_voltage=1)
```

Configures the instrument to apply a source current, which can take optional parameters that defer to the *source\_current\_range* and *compliance\_voltage* properties.

```
apply_voltage(max_voltage=1, compliance_current=0.01)
```

Configures the instrument to apply a source voltage, which can take optional parameters that defer to the *source\_voltage\_range* and *compliance\_current* properties.

#### property compliance\_current

Control the compliance current in Amps, which can take values from 5 to 120 mA.

7.68. Yokoqawa 637

#### property compliance\_voltage

Control the compliance voltage in Volts, which can take values between 1 and 30 V.

#### disable\_source()

Disables the source of current or voltage depending on the configuration of the instrument.

#### enable\_source()

Enables the source of current or voltage depending on the configuration of the instrument.

#### property id

Get the identification of the instrument

## ramp\_to\_current(current, steps=25, duration=0.5)

Ramps the current to a value in Amps by traversing a linear spacing of current steps over a duration, defined in seconds.

#### **Parameters**

- **steps** A number of linear steps to traverse
- duration A time in seconds over which to ramp

# ramp\_to\_voltage(voltage, steps=25, duration=0.5)

Ramps the voltage to a value in Volts by traversing a linear spacing of voltage steps over a duration, defined in seconds.

#### **Parameters**

- steps A number of linear steps to traverse
- **duration** A time in seconds over which to ramp

# shutdown()

Shuts down the instrument, and ramps the current or voltage to zero before disabling the source.

# property source\_current

Control the source current in Amps, if that mode is active. (float)

# property source\_current\_range

Control the current voltage range in Amps, which can take values: 1 mA, 10 mA, and 100 mA. Currents are truncated to an appropriate value if needed.

## property source\_enabled

Get a boolean value that is True if the source is enabled, determined by checking if the 5th bit of the OC flag is a binary 1.

## property source\_mode

Control the source mode, which can take the values 'current' or 'voltage'. The convenience methods <code>apply\_current()</code> and <code>apply\_voltage()</code> can also be used.

# property source\_voltage

Control the source voltage in Volts, if that mode is active. (float)

#### property source\_voltage\_range

Control the source voltage range in Volts, which can take values: 10 mV, 100 mV, 1 V, 10 V, and 30 V. Voltages are truncated to an appropriate value if needed.

# 7.68.2 Yokogawa GS200 Source

Bases: SCPIUnknownMixin. Instrument

Represents the Yokogawa GS200 source and provides a high-level interface for interacting with the instrument.

## property current\_limit

Control the current limit. "Limit" refers to maximum value of the electrical value that is conjugate to the mode (current is conjugate to voltage, and vice versa). Thus, current limit is only applicable when in 'voltage' mode

# property source\_enabled

Control whether the source is enabled. (bool)

# property source\_level

Control the output level, either a voltage or a current, depending on the source mode. (float)

# property source\_mode

Control the source mode. Can be either 'current' or 'voltage'.

## property source\_range

Control the range (either in voltage or current) of the output. "Range" refers to the maximum source level. (float)

# trigger\_ramp\_to\_level(level, ramp\_time)

Ramp the output level from its current value to "level" in time "ramp\_time". This method will NOT wait until the ramp is finished (thus, it will not block further code evaluation).

# **Parameters**

- level (float) final output level
- ramp\_time (float) time in seconds to ramp

## Returns

None

# property voltage\_limit

Control the voltage limit. "Limit" refers to maximum value of the electrical value that is conjugate to the mode (current is conjugate to voltage, and vice versa). Thus, voltage limit is only applicable when in 'current' mode

# 7.68.3 Yokogawa AQ6370 Series of Optical Spectrum Analyzers

Bases: SCPIMixin, Instrument

Represents Yokogawa AQ6370 Series of optical spectrum analyzer.

TRA

# Channel

Trace

7.68. Yokogawa 639

```
TRB
          Channel
               Trace
TRC
          Channel
               Trace
TRD
          Channel
               Trace
TRE
          Channel
               Trace
TRF
          Channel
               Trace
TRG
          Channel
               Trace
abort()
     Stop operations such as measurements and calibration.
property active_trace
     Control the active trace (str 'A', 'B', 'C', ...).
authenticate_ethernet(username, password=")
      Authenticate for an ethernet connection.
property automatic_sample_number
      Control the automatic sample number (bool).
property calc_result
     Get the results of the last analysis.
copy_trace(source, destination)
      Copy the data of specified trace to the another trace.
          Parameters
                 • source – Source trace (str 'A', 'B', 'C', ...).
                 • destination – Destination trace (str 'A', 'B', 'C', ...).
delete_trace(trace)
     Delete the specified trace.
          Parameters
               trace – Trace to be deleted (str 'ALL', 'A', 'B', 'C', ...).
execute_analysis()
     Execute the analysis with the current analysis settings.
```

#### get\_analysis()

Query the analysis results of latest analysis. If no analysis has been performed, returns query error.

#### get\_xdata(trace='TRA')

Measure the x-axis data of specified trace, output wavelength in m.

#### **Parameters**

**trace** – Trace to measure (str 'A', 'B', 'C', ...).

#### Returns

The x-axis data of specified trace.

# get\_ydata(trace='TRA')

Measure the y-axis data of specified trace, output power in dBm.

## **Parameters**

**trace** – Trace to measure (str 'A', 'B', 'C', ...).

#### Returns

The y-axis data of specified trace.

# initiate\_sweep()

Initiate a sweep.

# property level\_position

Control the reference level position regarding divisions (int, smaller than total number of divisions which is either 8, 10 or 12).(dynamic)

# property reference\_level

Control the reference level of main scale of level axis (float in dBm).

# property resolution\_bandwidth

Control the measurement resolution (float in m, discrete values: [0.02e-9, 0.05e-9, 0.1e-9, 0.2e-9, 0.5e-9, 1e-9, 2e-9] m).(dynamic)

## property sample\_number

Control the sample number (int from 51 to 50001).

## property sensitivity

Control the sweep sensitivity (str 'NHLD', 'NAUT', 'NORM', 'MID', 'HIGH1', 'HIGH2', 'HIGH3')

## set\_level\_position\_to\_max()

Set the reference level position to the maximum value.

#### property sweep\_complete

Get the completion status of the sweep (bool, True if complete).

# property sweep\_mode

Control the sweep mode (str 'SINGLE', 'REPEAT', 'AUTO', 'SEGMENT').

## property sweep\_time\_interval

Control the sweep time interval (int from 0 to 99999 s).

# property transfer\_format

Control the data transfer format. It returns to default ASCII at reset.

# trigger()

Perform a single sweep according to previous conditions.

7.68. Yokoqawa 641

wait\_for\_sweep\_complete(should\_stop=<function AQ6370Series.<lambda>>, timeout=300)
Block the program, waiting for the sweep to complete.

## **Parameters**

- **should\_stop** Function that returns True to stop waiting.
- **timeout** Maximum waiting time, in seconds.

#### Returns

True when sweep completed, False if stopped by should\_stop.

#### Raises

**TimeoutError** – If the temperature does not stabilize within the timeout period.

# property wavelength\_automatic\_center

Control whether the wavelength center follows the maximum (bool).

# property wavelength\_center

Control measurement condition center wavelength (float in m).(dynamic)

# property wavelength\_span

Control wavelength span (float from 0 to 1100e-9 m).(dynamic)

# property wavelength\_start

Control the measurement start wavelength (float from 50e-9 to 2250e-9 in m).(dynamic)

# property wavelength\_stop

Control the measurement stop wavelength (float from 50e-9 to 2250e-9 in m).(dynamic)

Bases: AQ6370Series

Represents Yokogawa AQ6370E optical spectrum analyzer.

TRA

# Channel

Trace

TRB

## Channel

Trace

TRC

#### Channel

Trace

TRD

## Channel

Trace

TRE

# Channel

Trace

TRF

Channel

Trace

TRG

Channel

Trace

#### property sensitivity\_level

Control the sweep sensitivity by specifying the sensitivity level you want to measure at, in dBm. The sensitivity closest to that level, and the sweep speed are automatically selected.

#### property sweep\_speed

Control the sweep speed (str '1x' or '2x' for double speed).

Bases: AQ6370Series

Represents Yokogawa AQ6370D optical spectrum analyzer.

TRA

Channel

Trace

TRB

Channel

Trace

TRC

Channel

Trace

TRD

Channel

Trace

TRE

Channel

Trace

TRF

Channel

Trace

TRG

Channel

Trace

#### property sweep\_speed

Control the sweep speed (str '1x' or '2x' for double speed).

7.68. Yokogawa 643

```
class pymeasure.instruments.yokogawa.aq6370series.AQ6370C(adapter, name='Yokogawa AQ3670D
                                                                OSA', baud_rate=115200, **kwargs)
     Bases: AQ6370Series
     Represents Yokogawa AQ6370C optical spectrum analyzer.
     TRA
               Channel
                   Trace
     TRB
               Channel
                   Trace
     TRC
               Channel
                   Trace
     TRD
               Channel
                   Trace
     TRE
               Channel
                   Trace
     TRF
               Channel
                   Trace
     TRG
               Channel
                   Trace
     property sweep_speed
          Control the sweep speed (str '1x' or '2x' for double speed).
class pymeasure.instruments.yokogawa.aq6370series.AQ6373(adapter, name='Yokogawa AQ3670D
                                                               OSA', baud_rate=115200, **kwargs)
     Bases: AQ6370Series
     Represents Yokogawa AQ6373 optical spectrum analyzer.
     TRA
               Channel
                   Trace
     TRB
               Channel
                   Trace
```

TRC Channel Trace TRD Channel Trace TRE Channel Trace TRF Channel Trace TRG Channel Trace class pymeasure.instruments.yokogawa.aq6370series.AQ6373B(adapter, name='Yokogawa AQ3670D OSA', baud\_rate=115200, \*\*kwargs) Bases: AQ6373 Represents Yokogawa AQ6373B variant optical spectrum analyzer. TRA Channel Trace TRB Channel Trace TRC Channel Trace TRD Channel Trace TRE Channel Trace TRF Channel

7.68. Yokogawa 645

Trace

```
TRG
               Channel
                   Trace
     property sweep_speed
          Control the sweep speed (str '1x' or '2x' for double speed).
class pymeasure.instruments.yokogawa.aq6370series.AQ6375(adapter, name='Yokogawa AQ3670D
                                                               OSA', baud_rate=115200, **kwargs)
     Bases: AQ6370Series
     Represents Yokogawa AQ6375 optical spectrum analyzer.
     TRA
               Channel
                   Trace
     TRB
               Channel
                   Trace
     TRC
               Channel
                   Trace
     TRD
               Channel
                   Trace
     TRE
               Channel
                   Trace
     TRF
               Channel
                   Trace
     TRG
               Channel
                   Trace
class pymeasure.instruments.yokogawa.aq6370series.AQ6375B(adapter, name='Yokogawa AQ3670D
                                                                OSA', baud_rate=115200, **kwargs)
     Bases: AQ6375
     Represents Yokogawa AQ6375B variant optical spectrum analyzer.
     TRA
               Channel
                   Trace
```

TRB

Channel

Trace

TRC

Channel Trace

Channel

Trace

TRE

TRD

Channel

Trace

TRF

Channel

Trace

TRG

Channel

Trace

#### property sweep\_speed

Control the sweep speed (str 1x or 2x for double speed).

7.68. Yokogawa 647

PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20250827				

**CHAPTER** 

**EIGHT** 

#### CONTRIBUTING

Contributions to the instrument repository and the main code base are highly encouraged. This section outlines the basic work-flow for new contributors.

# 8.1 Using the development version

New features are added to the development version of PyMeasure, hosted on GitHub. We use Git version control to track and manage changes to the source code. On Windows, we recommend using GitHub Desktop. Make sure you have an appropriate version of Git (or GitHub Desktop) installed and that you have a GitHub account.

Make sure git --version works from comand line. If you are using Github Desktop, you may need to add Git to your path in each terminal window. On windows anaconda prompt it would be set PATH="C:Users<USERNAME>AppDataLocalGitHubDesktopapp-<######>resourcesappgitcmd";%PATH%

In order to add your feature, you need to first fork PyMeasure. This will create a copy of the repository under your GitHub account.

The instructions below assume that you have set up Anaconda, as described in the *Quick Start guide* and describe the terminal commands necessary. If you are using GitHub Desktop, take a look through their documentation to understand the corresponding steps.

Clone your fork of PyMeasure your-github-username/pymeasure. In the following terminal commands replace your desired path and GitHub username.

```
cd /path/for/code
git clone https://github.com/your-github-username/pymeasure.git
```

If you had already installed PyMeasure using pip, make sure to uninstall it before continuing.

```
pip uninstall pymeasure
```

Install PyMeasure in the editable mode and select optional extras.

```
cd /path/for/code/pymeasure
pip install -e .[tests,docs]
```

The -e will allow you to edit the files of PyMeasure and see the changes reflected. The square brackets include extra groups that allow you to run tests and build the documentation locally. Make sure to reset your notebook kernel or Python console when doing so. Now you have your own copy of the development version of PyMeasure installed!

Depending on your Python installation you may get an error messages saying that the file setup.py is missing or similar. Updating pip may solve the problem.

```
python -m pip install pip --upgrade
```

# 8.2 Working on a new feature

We use branches in Git to allow multiple features to be worked on simultaneously, without causing conflicts. The master branch contains the stable development version. Instead of working on the master branch, you will create your own branch off the master and merge it back into the master when you are finished.

Create a new branch for your feature before editing the code. For example, if you want to add the new instrument "Extreme 5000" you will make a new branch "dev/extreme-5000".

```
git branch dev/extreme-5000
```

You can also make a new branch on GitHub. If you do so, you will have to fetch these changes before the branch will show up on your local computer.

```
git fetch
```

Once you have created the branch, change your current branch to match the new one.

```
git checkout dev/extreme-5000
```

Now you are ready to write your new feature and make changes to the code. To ensure consistency, please follow the *coding standards for PyMeasure*. Use git status to check on the files that have been changed. As you go, commit your changes and push them to your fork.

```
git add file-that-changed.py
git commit -m "A short description about what changed"
git push
```

# 8.3 Making a pull request

While you are working, it is helpful to start a pull request (PR) targeting the master branch of pymeasure/pymeasure. This will allow you to discuss your feature with other contributors. We encourage you to start this pull request already after your first commit. You may mark a pull request as a draft, if it is in an early state.

Start a pull request on the PyMeasure GitHub page.

There is some automation in place to run the unit tests and check some coding standards. Annotations in the "Files changed" tab indicate problems for you to correct (e.g. linting or docstring warnings).

Your pull-request will be reviewed by the PyMeasure maintainers. Frequently there is some iteration and discussion based on that feedback until a pull request can be merged. This will happen either in the conversation tab or in inline code comments.

Be aware that due to maintainer manpower limitations it might take a long time until PRs get reviewed and/or merged. In general, review effort scales badly with PR size. Therefore, **smaller PRs are much preferred**. Try to limit your contribution to one "aspect", e.g. one instrument (or a few if closely related), one bug fix, or one feature contribution.

If you placed your contribution in a separate branch as suggested above, you can easily use your contribution in the meantime – just check out your feature branch instead of *master*.

# 8.4 Unit testing

Unit tests are run each time a new commit is made to a branch. The purpose is to catch changes that break the current functionality, by testing each feature unit. PyMeasure relies on pytest to preform these tests, which are run on TravisCI and Appveyor for Linux/macOS and Windows respectively.

Running the unit tests while you develop is highly encouraged. This will ensure that you have a working contribution when you create a pull request.

#### pytest

If your feature can be tested, unit tests are required. This will ensure that your features keep working as new features are added.

Now you are familiar with all the pieces of the PyMeasure development work-flow. We look forward to seeing your pull-request!

8.4. Unit testing 651



CHAPTER
NINE

# **REPORTING AN ERROR**

Please report all errors to the Issues section of the PyMeasure GitHub repository. Use the search function to determine if there is an existing or resolved issued before posting.

PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20	250827	
CFA	01 1 0	

**CHAPTER** 

**TEN** 

#### **ADDING INSTRUMENTS**

You can make a significant contribution to PyMeasure by adding a new instrument to the pymeasure.instruments package. Even adding an instrument with a few features can help get the ball rolling, since its likely that others are interested in the same instrument.

Before getting started, become familiar with the *contributing work-flow* for PyMeasure, which steps through the process of adding a new feature (like an instrument) to the development version of the source code.

Pymeasure instruments communicate with the devices via transfer of bytes or ASCII characters encoded as bytes. For ease of use, we have *property creators* to easily create python properties. Similarly, we have creators to easily implement *channels*. Finally, for a smoother implementation process and better maintenance, we have *tests*.

The following sections will describe how to lay out your instrument code.

# 10.1 File structure

Your new instrument should be placed in the directory corresponding to the manufacturer of the instrument. For example, if you are going to add an "Extreme 5000" instrument you should add the following files assuming "Extreme" is the manufacturer. Use lowercase for all filenames to distinguish packages from CamelCase Python classes.

## 10.1.1 Updating the init file

The \_\_init\_\_.py file in the manufacturer directory should import all of the instruments that correspond to the manufacturer, to allow the files to be easily imported.

#### 10.1.2 Add test files

Test files (pytest) for each instrument are highly encouraged, as they help verify the code and implement changes. Testing new code parts with a test (Test Driven Development) is a good way for fast and good programming, as you catch errors early on.

## 10.1.3 Adding documentation

Documentation for each instrument is required, and helps others understand the features you have implemented. Add a new reStructuredText file to the documentation.

Copy an existing instrument documentation file, which will automatically generate the documentation for the instrument. The index.rst file should link to the extreme5000 file. For a new manufacturer, the manufacturer should be also linked in pymeasure/docs/api/instruments/index.rst.

#### 10.2 Instrument file

All standard instruments should be child class of *Instrument*. This provides the basic functionality for working with *Adapters*, which perform the actual communication.

The most basic instrument, for our "Extreme 5000" example starts like this:

```
# This file is part of the PyMeasure package.
# Copyright (c) 2013-2025 PyMeasure Developers
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY. WHETHER IN AN ACTION OF CONTRACT. TORT OR OTHERWISE. ARISING FROM.
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
from pymeasure.instruments import Instrument
```

This is a minimal instrument definition:

```
adapter,
name,
**kwargs
)
```

Make sure to include the PyMeasure license to each file, and add yourself as an author to the AUTHORS.txt file.

There is a certain order of elements in an instrument class that is useful to adhere to:

- First, the initializer (the \_\_init\_\_() method), this makes it faster to find when browsing the source code.
- Then class attributes/variables, if you need them.
- Then properties (pymeasure-specific or generic Python variants). This will be the bulk of the implementation.
- · Finally, any methods.

## 10.3 Your instrument's user interface

Your instrument will have a certain set of properties and methods that are available to a user and discoverable via the documentation or their editor's autocomplete function.

In principle you are free to choose how you do this (with the exception of standard SCPI properties like id). However, there are a couple of practices that have turned out to be useful to follow:

- Naming things is important. Try to choose clear, expressive, unambiguous names for your instrument's elements.
- If there are already similar instruments in the same "family" (like a power supply) in pymeasure, try to follow their lead where applicable. It's better if, e.g., all power supplies have a current\_limit instead of an assortment of current\_max, Ilim, max\_curr, etc.
- If there is already an instrument with a similar command set, check if you can inherit from that one and just tweak a couple of things. This massively reduces code duplication and maintenance effort. The section *Instruments with similar features* shows how to achieve that.
- The bulk of your instrument's interface will probably be made up of properties for quantities to set and/or read out. Our custom properties (see *Writing properties* ff. below) offer some convenience features and are therefore preferable, but plain Python properties are also fine.
- "Actions", commands or verbs should typically be methods, not properties: recall(), trigger\_scan(), prepare\_resistance\_measurement(), etc.
- This separation between properties and methods also naturally helps with observing the "command-query separation" principle.
- If your instrument has multiple identical channels, see *Instruments with channels*.

In principle, you are free to write any methods that are necessary for interacting with the instrument. When doing so, make sure to use the self.ask(command), self.write(command), and self.read() methods to issue commands instead of calling the adapter directly. If the communication requires changes to the commands sent/received, you can override these methods in your instrument, for further information see *Advanced communication protocols*.

In practice, we have developed a number of best practices for making instruments easy to write and maintain. The following sections detail these, which are highly encouraged to follow.

## 10.3.1 Common instrument types

There are a number of categories that many instruments fit into. In the future, pymeasure should gain an abstraction layer based on that, see this issue. Until that is ready, here are a couple of guidelines towards a more uniform API. Note that not all already available instruments follow these, but expect this to be harmonized in the future.

#### **Generic types mixins**

The *generic\_types* module contains mixin classes for common types. For example, if an instrument complies to SCPI standards, you can add *SCPIMixin* to your instrument:

```
from pymeasure.instruments.generic_types import SCPIMixin

class SomeSCPIInstrument(SCPIMixin, Instrument):
    """This instrument has properties and methods defined for all SCPI instruments"""
```

This mixin adds default SCPI properties like id, status and default methods like clear() and reset() to SomeSCPIInstrument.

#### **Frequent properties**

If your instrument has an **output** that can be switched on and off, use a boolean property called output\_enabled.

#### **Power supplies**

PSUs typically can measure the *actual* current and voltage, as well as have settings for the voltage level and the current limit. To keep naming clear and avoid confusion, implement the properties current, voltage, voltage\_setpoint and current\_limit, respectively.

#### 10.3.2 Managing status codes or other indicator values

Often, an instrument features one or more collections of specific values that signal some status, an instrument mode or a number of possible configuration values. Typically, these are collected in mappings of some sort, as you want to provide a clear and understandable value to the user, while abstracting away the raw data, think ACQUISITION\_MODE instead of 0x04. The mappings normally are kept at module level (i.e. not defined within the instrument class), so that they are available when using the property factories. This is a small drawback of using Python class attributes.

The easiest way to handle these mappings is a plain dict. However, there is often a better way, the Python enum. Enum. To cite the Python documentation,

An Enum is a set of symbolic names bound to unique values. They are similar to global variables, but they offer a more useful repr(), grouping, type-safety, and a few other features.

As our signal values are often integers, the most appropriate enum types are IntEnum and IntFlag.

IntEnum is the same as Enum, but its members are also integers and can be used anywhere that an integer can be used (so their use for composing commands is transparent), but logic/code they appear in is much more legible. Note that starting from Python version 3.11, the printed format of the IntEnum and IntFlag has been changed to return numeric value; however, the symbolic name can be obtained by printing its repr or the .name property, or returning the value in a REPL.

```
>>> from enum import IntEnum
>>> class InstrMode(IntEnum):
        WAITING = 0x00
        HEATING = 0x01
        COOLING = 0x05
>>> received_from_device = 0x01
>>> current_mode = InstrMode(received_from_device)
>>> if current_mode == InstrMode.WAITING:
        print('Idle')
. . .
... else:
        current_mode
. . .
        print(repr(current_mode))
        print(f'Mode value: {current_mode}')
<InstrMode.HEATING: 1>
<InstrMode.HEATING: 1>
Mode value: 1
```

IntFlag has the added benefit that it supports bitwise operators and combinations, and as such is a good fit for status bitmasks or error codes that can represent multiple values:

IntFlags are used by many instruments for the purpose just demonstrated.

The status property could look like this:

```
status = Instrument.measurement(
    "STB?",
    """Measure the status of the device as enum.""",
    get_process=lambda v: ErrorCode(v),
)
```

# 10.4 Defining default connection settings

When implementing instruments, it's sometimes necessary to define default connection settings. This might be because an instrument connection requires *specific non-default settings*, or because your instrument actually supports *multiple interfaces*.

The VISAAdapter class offers a flexible way of dealing with connection settings fully within the initializer of your instrument.

# 10.4.1 Single interface

The simplest version, suitable when the instrument connection needs default settings, just passes all keywords through to the Instrument initializer, which hands them over to *VISAAdapter* if adapter is a string or integer.

```
def __init__(self, adapter, name="Extreme 5000", **kwargs):
    super().__init__(
        adapter,
        name,
        **kwargs
)
```

If you want to set defaults that should be prominently visible to the user and may be overridden, place them in the signature. This is suitable when the instrument has one type of interface, or any defaults are valid for all interface types, see the documentation in *VISAAdapter* for details.

```
def __init__(self, adapter, name="Extreme 5000", baud_rate=2400, **kwargs):
    super().__init__(
        adapter,
        name,
        baud_rate=baud_rate,
        **kwargs
)
```

If you want to set defaults, but they don't need to be prominently exposed for replacement, use this pattern, which sets the value only when there is no entry in kwargs, yet.

```
def __init__(self, adapter, name="Extreme 5000", **kwargs):
    kwargs.setdefault('timeout', 1500)
    super().__init__(
        adapter,
        name,
        **kwargs
    )
```

## 10.4.2 Multiple interfaces

Now, if you have instruments with multiple interfaces (e.g. serial, TCPI/IP, USB), things get interesting. You might have settings common to all interfaces (like timeout), but also settings that are only valid for one interface type, but not others.

The trick is to add keyword arguments that name the interface type, like asrl or gpib, below (see here for the full list). These then contain a *dictionary* with the settings specific to the respective interface:

When the instrument instance is created, the interface-specific settings for the actual interface being used get merged with \*\*kwargs before passing them on to PyVISA, the rest is discarded. This way, we always pass on a valid set of arguments. In addition, any entries in \*\*kwargs\*\* take precedence, so if they need to, it is *still* possible for users to override any defaults you set in the instrument definition.

For many instruments, the simple way presented first is enough, but in case you have a more complex arrangement to implement, see whether *Advanced communication protocols* fits your bill. If, for some exotic reason, you need a special connection type, which you cannot model with PyVISA, you can write your own Adapter.

# 10.5 Writing properties

In PyMeasure, Python properties are the preferred method for dealing with variables that are read or set.

## 10.5.1 The property factories

PyMeasure comes with three central convenience factory functions for making properties for classes: *CommonBase.control*, *CommonBase.measurement*, and *CommonBase.setting*. You can call them, however, as Instrument.control, Instrument.measurement, and Instrument.setting.

The *Instrument.measurement* function returns a property that can only read values from an instrument. For example, if our "Extreme 5000" has the :TEMP? command, we can write the following property to be added after the def \_\_init\_\_ line in our above example class, or added to the class after the fact as in the code here:

```
Extreme5000.cell_temp = Instrument.measurement(
   ":TEMP?",
   """Measure the temperature of the reaction cell.""",
)
```

You will notice that a documentation string is required, see *Docstrings* for details.

When we use this property we will get the temperature of the reaction cell.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.cell_temp # Sends ":TEMP?" to the device
127.2
```

The *Instrument.control* function extends this behavior by creating a property that you can read and set. For example, if our "Extreme 5000" has the :VOLT? and :VOLT <float> commands that are in Volts, we can write the following property.

```
Extreme5000.voltage = Instrument.control(
    ":VOLT?", ":VOLT %g",
    """Control the voltage in Volts (float)."""
)
```

You will notice that we use the Python string format %g to format passed-through values as floating point.

We can use this property to set the voltage to 100 mV, which will send the appropriate command, and then to request the current voltage:

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 0.1  # Sends ":VOLT 0.1" to set the voltage to 100 mV
>>> extreme.voltage  # Sends ":VOLT?" to query for the current value
0.1
```

Finally, the *Instrument.setting* function can only set, but not read values.

Using the *Instrument.control*, *Instrument.measurement*, and *Instrument.setting* functions, you can create a number of properties for basic measurements and controls.

The next sections detail additional features of the property factories. These allow you to write properties that cover specific ranges, or that have to map between a real value to one used in the command. Furthermore it is shown how to perform more complex processing of return values from your device.

## 10.5.2 Restricting values with validators

Many GPIB/SCPI commands are more restrictive than our basic examples above. The *Instrument.control* function has the ability to encode these restrictions using *validators*. A validator is a function that takes a value and a set of values, and returns a valid value or raises an exception. There are a number of pre-defined validators in *pymeasure*. *instruments.validators* that should cover most situations. We will cover the four basic types here.

In the examples below we assume you have imported the validators.

In many situations you will also need to process the return string in order to extract the wanted quantity or process a value before sending it to the device. The *Instrument.control*, *Instrument.measurement* and *Instrument.setting* functions also provide means to achieve this.

#### In a restricted range

If you have a property with a restricted range, you can use the strict\_range and truncated\_range functions.

For example, if our "Extreme 5000" can only support voltages from -1 V to 1 V, we can modify our previous example to use a strict validator over this range.

```
Extreme5000.voltage = Instrument.control(
    ":VOLT?", ":VOLT %g",
    """Control the voltage in Volts (float strictly from -1 to 1).""",
    validator=strict_range,
    values=[-1, 1]
)
```

Now our voltage will raise a ValueError if the value is out of the range.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 100
Traceback (most recent call last):
...
ValueError: Value of 100 is not in range [-1,1]
```

This is useful if you want to alert the programmer that they are using an invalid value. However, sometimes it can be nicer to truncate the value to be within the range.

```
Extreme5000.voltage = Instrument.control(
    ":VOLT?", ":VOLT %g",
    """Control the voltage in Volts (float from -1 to 1).

Invalid voltages are truncated.
    """,
    validator=truncated_range,
    values=[-1, 1]
)
```

Now our voltage will not raise an error, and will truncate the value to the range bounds.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 100  # Executes ":VOLT 1"
>>> extreme.voltage
1.0
```

#### In a discrete set

Often a control property should only take a few discrete values. You can use the strict\_discrete\_set and truncated\_discrete\_set functions to handle these situations. The strict version raises an error if the value is not in the set, as in the range examples above.

For example, if our "Extreme 5000" has a :RANG <float> command that sets the voltage range that can take values of 10 mV, 100 mV, and 1 V in Volts, then we can write a control as follows.

```
validator=truncated_discrete_set,
  values=[10e-3, 100e-3, 1]
)
```

Now we can set the voltage range, which will automatically truncate to an appropriate value.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 0.08
>>> extreme.voltage
0.1
```

## 10.5.3 Mapping values

Now that you are familiar with the validators, you can additionally use maps to satisfy instruments which require non-physical values. The map\_values argument of <code>Instrument.control</code> enables this feature.

If your set of values is a list, then the command will use the index of the list. For example, if our "Extreme 5000" instead has a :RANG <integer>, where 0, 1, and 2 correspond to 10 mV, 100 mV, and 1 V, then we can use the following control.

```
Extreme5000.voltage = Instrument.control(
    ":RANG?", ":RANG %d",
    """Control the voltage range in Volts (float in 10 mV, 100 mV and 1 V).
    """,
    validator=truncated_discrete_set,
    values=[10e-3, 100e-3, 1],
    map_values=True
)
```

Now the actual GPIB/SCIP command is ":RANG 1" for a value of 100 mV, since the index of 100 mV in the values list is 1.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 100e-3
>>> extreme.read()
'1'
>>> extreme.voltage = 1
>>> extreme.voltage
1
```

Dictionaries provide a more flexible method for mapping between real-values and those required by the instrument. If instead the :RANG <integer> took 1, 2, and 3 to correspond to 10 mV, 100 mV, and 1 V, then we can replace our previous control with the following.

```
Extreme5000.voltage = Instrument.control(
    ":RANG?", ":RANG %d",
    """Control the voltage range in Volts (float in 10 mV, 100 mV and 1 V).
    """,
    validator=truncated_discrete_set,
    values={10e-3:1, 100e-3:2, 1:3},
    map_values=True
)
```

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 10e-3
>>> extreme.read()
'1'
>>> extreme.voltage = 100e-3
>>> extreme.voltage
0.1
```

The dictionary now maps the keys to specific values. The values and keys can be any type, so this can support properties that use strings:

```
Extreme5000.channel = Instrument.control(
    ":CHAN?", ":CHAN %d",
    """Control the measurement channel (string strictly in 'X', 'Y', 'Z').""",
    validator=strict_discrete_set,
    values={'X':1, 'Y':2, 'Z':3},
    map_values=True
)
```

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.channel = 'X'
>>> extreme.read()
'1'
>>> extreme.channel = 'Y'
>>> extreme.channel
'Y'
```

As you have seen, the *Instrument.control* function can be significantly extended by using validators and maps.

## 10.5.4 Boolean properties

The idea of using maps can be leveraged to implement properties where the user-facing values are booleans, so you can interact in a pythonic way using True and False:

```
Extreme5000.output_enabled = Instrument.control(
    "OUTP?", "OUTP %d",
    """Control the instrument output is enabled (boolean).""",
    validator=strict_discrete_set,
    map_values=True,
    values={True: 1, False: 0}, # the dict values could also be "on" and "off", etc...
    depending on the device
)
```

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.output_enabled = True
>>> extreme.read()
'1'
>>> extreme.output_enabled = False
>>> extreme.output_enabled
False
>>> # Invalid input raises an exception
>>> extreme.output_enabled = 34
```

```
Traceback (most recent call last):
...
ValueError: Value of 34 is not in the discrete set {True: 1, False: 0}
```

Good names for boolean properties are chosen such that they could also be a yes/no question: "Is the output enabled?" -> output\_enabled, display\_active, etc.

# 10.5.5 Processing of set values

The *Instrument.control*, and *Instrument.setting* allow a keyword argument *set\_process* which must be a function that takes a value after validation and performs processing before value mapping. This function must return the processed value. This can be typically used for unit conversions as in the following example:

```
Extreme5000.current = Instrument.setting(
    ":CURR %g",
    """Set the measurement current in A (float strictly from 0 to 10).""",
    validator=strict_range,
    values=[0, 10],
    set_process=lambda v: 1e3*v, # convert current to mA
)
```

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.current = 1 # set current to 1000 mA
```

# 10.5.6 Processing of return values

Similar to *set\_process* the *Instrument.control*, and *Instrument.measurement* functions allow a *get\_process* argument which if specified must be a function that takes a value and performs processing before value mapping. The function must return the processed value. In analogy to the example above this can be used for example for unit conversion:

```
Extreme5000.current = Instrument.control(
    ":CURR?", ":CURR %g",
    """Control the measurement current in A (float strictly from 0 to 10).""",
    validator=strict_range,
    values=[0, 10],
    set_process=lambda v: 1e3*v, # convert to mA
    get_process=lambda v: 1e-3*v, # convert to A
)
```

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.current = 3.1
>>> extreme.current
3.1
```

Another use-case of *set-process*, *get-process* is conversion from/to a pint.Quantity. Modifying above example to set or return a quantity, we get:

```
from pymeasure.units import ureg

(continues on next page)
```

```
Extreme5000.current = Instrument.control(
    ":CURR?", ":CURR %g",
    """Control the measurement current (float).""",
    set_process=lambda v: v.m_as(ureg.mA), # send the value as mA to the device
    get_process=lambda v: ureg.Quantity(v, ureg.mA), # convert to quantity
)
```

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.current = 3.1 * ureg.A
>>> extreme.current.m_as(ureg.A)
3.1
```

#### 1 Note

This is, how quantities can be used in pymeasure instruments right now. Issue 666 develops a more convenient implementation of quantities in the property factories.

*get\_process* can also be used to perform string processing. Let's say your instrument returns a value with its unit (e.g. 1.23 nF), which has to be removed. This could be achieved by the following code:

```
Extreme5000.capacity = Instrument.measurement(
    ":CAP?",
    """Measure the capacity in nF (float).""",
    get_process=lambda v: float(v.replace('nF', ''))
)
```

The same can be also achieved by the *preprocess\_reply* keyword argument to *Instrument.control* or *Instrument.* measurement. This function is forwarded to Adapter.values and runs directly after receiving the reply from the device. One can therefore take advantage of the built in casting abilities and simplify the code accordingly:

```
Extreme5000.capacity = Instrument.measurement(
    ":CAP?",
    """Measure the capacity in nF (float).""",
    preprocess_reply=lambda v: v.replace('nF', '')
    # notice how we don't need to cast to float anymore
)
```

# 10.5.7 Checking the instrument for errors

If you need to separately ask your instrument about its error state after getting/setting, use the parameters check\_get\_errors and check\_set\_errors of control(), respectively. If those are enabled, the methods check\_get\_errors() and check\_set\_errors(), respectively, will be called after device communication has concluded. In the default implementation, for simplicity both methods call check\_errors(). To read the automatic response of instruments that respond to every set command with an acknowledgment or error, override check\_set\_errors() as needed.

## 10.5.8 Using multiple values

Seldomly, you might need to send/receive multiple values in one command. The *Instrument.control* function can be used with multiple values at one time, passed as a tuple. Say, we may set voltages and frequencies in our "Extreme 5000", and the commands for this are :VOLTFREQ? and :VOLTFREQ <float>,<float>, we could use the following property:

```
Extreme5000.combination = Instrument.control(
    ":VOLTFREQ?", ":VOLTFREQ %g,%g",
    """Simultaneously control the voltage in Volts and the frequency in Hertz (both.

→float).

This property is set by a tuple.
    """
)
```

In use, we could set the voltage to 200 mV, and the Frequency to 931 Hz, and read both values immediately afterwards.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.combination = (0.2, 931)  # Executes ":VOLTFREQ 0.2,931"
>>> extreme.combination  # Reads ":VOLTFREQ?"

[0.2, 931.0]
```

This interface is not too convenient, but luckily not often needed.

# 10.5.9 Dynamic properties

As described in previous sections, Python properties are a very powerful tool to easily code an instrument's programming interface. One very interesting feature provided in PyMeasure is the ability to adjust properties' behaviour in subclasses or dynamically in instances. This feature allows accommodating some interesting use cases with a very compact syntax.

Dynamic features of a property are enabled by setting its dynamic parameter to True.

Afterwards, creating specifically-named attributes (either in class definitions or on instances) allows modifying the parameters used at the time of property definition. You need to define an attribute whose name is *property name>\_property\_parameter> and assign to it the desired value. Pay attention <i>not* to inadvertently define other class attribute or instance attribute names matching this pattern, since they could unintentionally modify the property behaviour.



To clearly distinguish these special attributes from normal class/instance attributes, they can only be set, not read.

The mechanism works for all the parameters in properties, except dynamic and docs – see *Instrument.control*, *Instrument.measurement*, *Instrument.setting*.

#### Dynamic validity range

Let's assume we have an instrument with a command that accepts a different valid range of values depending on its current state. The code below shows how this can be accomplished with dynamic properties.

```
Extreme5000.voltage = Instrument.control(
    ": VOLT?", ": VOLT %g",
    """Control the voltage in Volts (float).""",
   validator=strict_range,
   values=[-1, 1],
   dvnamic = True.
def set_bipolar_mode(self, enabled = True):
    """Safely switch between bipolar/unipolar mode."""
    # some code to switch off the output first
    # ...
   if enabled:
        self.mode = "BIPOLAR"
        # set valid range of "voltage" property
        self.voltage\_values = [-1, 1]
   else:
        self.mode = "UNIPOLAR"
        # note the "propertyname_parametername" form of the attribute
        self.voltage_values = [0, 1]
```

Now our voltage property has a dynamic validity range, either [-1, 1] or [0, 1]. A side effect of this is that the property's docstring should be less specific, to avoid it containing dynamically changed information (like the admissible value range). In this example, the property name was voltage and the parameter to adjust was values, so we used self. voltage\_values to set our desired values.

## 10.6 Instruments with similar features

When instruments have a similar set of features, it makes sense to use inheritance to obtain most of the functionality from a parent instrument class, instead of copy-pasting code.



Don't forget to update the instrument's name attribute accordingly, by either supplying an appropriate argument (if available) during the super().\_\_init\_\_() call, or by setting it anew below that call.

In some cases, one only needs to add additional properties and methods. In other cases, some of the already present properties/methods need to be completely replaced by defining them again in the derived class. Often, however, only some details need to be changed. This can be dealt with efficiently using dynamic properties.

## 10.6.1 Instrument family with different parameter values

A common case is to have a family of similar instruments with some parameter range different for each family member. In this case you would update the specific class parameter range without rewriting the entire property:

```
class FictionalInstrumentFamily(Instrument):
    frequency = Instrument.setting(
        "FREQ %g",
        """Set the frequency (float).""",
        validator=strict_range,
        values=[0, 1e9],
        dynamic=True,
        # ... other possible parameters follow
   )
    #
      ... complete class implementation here
class FictionalInstrument_1GHz(FictionalInstrumentFamily):
class FictionalInstrument_3GHz(FictionalInstrumentFamily):
    frequency_values = [0, 3e9]
class FictionalInstrument_9GHz(FictionalInstrumentFamily):
    frequency_values = [0, 9e9]
```

Notice how easily you can derive the different family members from a common class, and the fact that the attribute is now defined at class level and not at instance level.

## 10.6.2 Instruments with similar command syntax

Another use case involves maintaining compatibility between instruments with commands having different syntax, like in the following example.

```
class MultimeterA(Instrument):
    voltage = Instrument.measurement(get_command="VOLT?",...)

# ...full class definition code here

class MultimeterB(MultimeterA):
    # Same as brand A multimeter, but the command to read voltage
    # is slightly different
    voltage_get_command = "VOLTAGE?"
```

In the above example, MultimeterA and MultimeterB use a different command to read the voltage, but the rest of the behaviour is identical. MultimeterB can be defined subclassing MultimeterA and just implementing the difference.

# 10.7 Instruments with channels

Some instruments, like oscilloscopes and voltage sources, have channels whose commands differ only in the channel name. For this case, we have *Channel*, which is similar to *Instrument* and its property factories, but does expect an *Instrument* instance (i.e., a parent instrument) instead of an *Adapter* as parameter. All the channel communication is routed through the instrument's methods (*write*, *read*, etc.). However, *Channel.insert\_id* uses str.format to insert the channel's id at any occurrence of the class attribute Channel.placeholder, which defaults to "ch", in the written commands. For example "Ch{ch}:VOLT?" will be sent as "Ch3:VOLT?" to the device, if the channel's id is "3".

Please add any created channel classes to the documentation. In the instrument's documentation file, you may add

```
.. autoclass:: pymeasure.instruments.MANUFACTURER.INSTRUMENT.CHANNEL
:members:
:show-inheritance:
```

MANUFACTURER is the folder name of the manufacturer and INSTRUMENT the file name of the instrument definition, which contains the CHANNEL class. You may link in the instrument's docstring to the channel with :class:`CHANNEL`

To simplify and standardize the creation of channels in an Instrument class, there are two classes that can be used. For instruments with fewer than 16 channels, *ChannelCreator* should be used to explicitly declare each individual channel. For instruments with more than 16 channels, the *MultiChannelCreator* can create multiple channels in a single declaration.

## 10.7.1 Adding a channel with Channel Creator

For instruments with fewer than 16 channels the class *ChannelCreator* should be used to assign each channel interface to a class attribute. *ChannelCreator* constructor accepts two parameters, the channel class for this channel interface, and the instrument's channel id for the channel interface.

In this example, we are defining a channel class and an instrument driver class. The VoltageChannel channel class will be used for controlling two channels in our ExtremeVoltage5000 instrument. In the ExtremeVoltage5000 class we declare two class attributes with *ChannelCreator*, output\_A and output\_B, which will become our channel interfaces.

```
class VoltageChannel(Channel):
    """A channel of the voltage source."""

voltage = Channel.control(
    "SOURce{ch}:VOLT?", "SOURce{ch}:VOLT %g",
    """"Control the output voltage of this channel.""",
)

class ExtremeVoltage5000(Instrument):
    """An instrument with channels."""
    output_A = Instrument.ChannelCreator(VoltageChannel, "A")
    output_B = Instrument.ChannelCreator(VoltageChannel, "B")
```

At instrument class instantiation, the instrument class will create an instance of the channel class and assign it to the class attribute name. Additionally the channels will be collected in a dictionary, by default named channels. We can access the channel interface through that class name:

```
extreme_inst = ExtremeVoltage5000('COM3')
# Set channel A voltage
extreme_inst.output_A.voltage = 50
# Read channel B voltage
chan_b_voltage = extreme_inst.output_B.voltage
```

Or we can access the channel interfaces through the channels collection:

```
# Set channel A voltage
extreme_inst.channels['A'].voltage = 50
# Read channel B voltage
chan_b_voltage = extreme_inst.channels['B'].voltage
```

## 10.7.2 Adding multiple channels with MultiChannelCreator

For instruments greater than 16 channels the class *MultiChannelCreator* can be used to easily generate a list of channels from one class attribute declaration.

The *MultiChannelCreator* constructor accepts a single channel class or list of channel classes, and a list of corresponding channel ids. Instead of lists, you may also use tuples. If you give a single class and a list of ids, all channels will be of the same class.

At instrument instantiation, the instrument will generate channel interfaces as class attribute names composing of the prefix (default "ch\_") and channel id, e.g. the channel with id "A" will be added as attribute ch\_A. While ChannelCreator creates a channel interface for each class attribute, MultiChannelCreator creates a channel collection for the assigned class attribute. It is recommended you use the class attribute name channels to keep the codebase homogenous.

To modify our example, we will use *MultiChannelCreator* to generate 24 channels of the VoltageChannel class.

```
class VoltageChannel(Channel):
    """A channel of the voltage source."""

voltage = Channel.control(
    "SOURce{ch}:VOLT?", "SOURce{ch}:VOLT %g",
    """Control the output voltage of this channel.""",
)

class MultiExtremeVoltage5000(Instrument):
    """An instrument with channels."""
    channels = Instrument.MultiChannelCreator(VoltageChannel, list(range(1,25)))
```

We can now access the channel interfaces through the generated class attributes:

```
extreme_inst = MultiExtremeVoltage5000('COM3')
# Set channel 5 voltage
extreme_inst.ch_5.voltage = 50
# Read channel 16 voltage
chan_16_voltage = extreme_inst.ch_16.voltage
```

Because we use *channels* as the class attribute for MultiChannelCreator, we can access the channel interfaces through the channels collection:

```
# Set channel 10 voltage
extreme_inst.channels[10].voltage = 50
# Read channel 22 voltage
chan_b_voltage = extreme_inst.channels[22].voltage
```

## 10.7.3 Advanced channel management

#### Adding / removing channels

In order to add or remove programmatically channels, use the parent's add\_child(), remove\_child() methods.

#### Channels with fixed prefix

If all channel communication is prefixed by a specific command, e.g. "SOURceA:" for channel A, you can override the channel's insert\_id() method. That is especially useful, if you have only one channel of that type, e.g. because it defines one function of the instrument vs. another one.

```
class VoltageChannelPrefix(Channel):
    """A channel of a voltage source, every command has the same prefix."""

def insert_id(self, command):
    return f"SOURce{self.id}:{command}"

voltage = Channel.control(
    "VOLT?", "VOLT %g",
    """Control the output voltage of this channel.""",
)
```

This channel class implements the same communication as the previous example, but implements the channel prefix in the insert\_id() method and not in the individual property (created by control()).

#### Collections of different channel types

Some devices have different types of channels. In this case, you can specify a different collection and prefix parameter.

```
class PowerChannel(Channel):
    """A channel controlling the power."""
    power = Channel.measurement(
        "POWER?", """Measure the currently consumed power.""")

class MultiChannelTypeInstrument(Instrument):
    """An instrument with two different channel types."""
    analog = Instrument.MultiChannelCreator(
        (VoltageChannel, VoltageChannelPrefix),
        ("A", "B"),
        prefix="an_")
    digital = Instrument.MultiChannelCreator(VoltageChannel, (0, 1, 2), prefix="di_")
    power = Instrument.ChannelCreator(PowerChannel)
```

This instrument has two collections of channels and one single channel. The first collection in the dictionary analog contains an instance of VoltageChannel with the name an\_A and an instance of VoltageChannelPrefix with the name an\_B. The second collection contains three channels of type VoltageChannel with the names di\_0, di\_1, di\_2 in the dictionary digital. You can address the first channel of the second group either with inst.di\_0 or with inst.digital[0]. Finally, the instrument has a single channel with the name power, as it does not have a prefix.

If you have a single channel category, do not change the default parameters of *ChannelCreator* or *add\_child()*, in order to keep the code base homogeneous. We expect the default behaviour to be sufficient for most use cases.

# 10.8 Advanced communication protocols

Some devices require a more advanced communication protocol, e.g. due to checksums or device addresses. In most cases, it is sufficient to subclass *Instrument.write* and *Instrument.read*.

# 10.8.1 Instrument's inner workings

In order to adjust an instrument for more complicated protocols, it is key to understand the different parts.

The Adapter exposes write() and read() for strings, write\_bytes() and read\_bytes() for bytes messages. These are the most basic methods, which log all the traffic going through them. For the actual communication, they call private methods of the Adapter in use, e.g. VISAAdapter.\_read. For binary data, like waveforms, the adapter provides also write\_binary\_values() and read\_binary\_values(), which use the aforementioned methods. You do not need to call all these methods directly, instead, you should use the methods of Instrument with the same name. They call the Adapter for you and keep the code tidy.

Now to *Instrument*. The most important methods are <code>write()</code> and <code>read()</code>, as they are the most basic building blocks for the communication. The pymeasure properties (<code>Instrument.control</code> and its derivatives <code>Instrument.measurement</code> and <code>Instrument.setting()</code> and probably most of your methods and properties will call them. In any instrument, <code>write()</code> should write a general string command to the device in such a way, that it understands it. Similarly, <code>read()</code> should return a string in a general fashion in order to process it further.

The getter of *Instrument.control* does not call them directly, but via a chain of methods. It calls values() which in turn calls ask() and processes the returned string into understandable values. ask() sends the readout command via write(), waits some time if necessary via wait\_for(), and reads the device response via read().

Similarly, Instrument.binary\_values sends a command via write(), waits with wait\_till\_read(), but reads the response via *Adapter.read\_binary\_values*.

# 10.8.2 Adding a device address and adding delay

Let's look at a simple example for a device, which requires its address as the first three characters and returns the same style. This is straightforward, as write() just prepends the device address to the command, and read() has to strip it again doing some error checking. Similarly, a checksum could be added. Additionally, the device needs some time after it received a command, before it responds, therefore wait\_for() waits always a certain time span.

```
class ExtremeCommunication(Instrument):
    """Control the ExtremeCommunication instrument.

    :param address: The device address for the communication.
    :param query_delay: Wait time after writing and before reading in seconds.
    """
    def __init__(self, adapter, name="ExtremeCommunication", address=0, query_delay=0.1):
```

```
super().__init__(adapter, name)
       self.address = f"{address:03}"
       self.query_delay = query_delay
   def write(self, command):
       """Add the device address in front of every command before sending it."""
       super().write(self.address + command)
   def wait_for(self, query_delay=0):
       """Wait for some time.
       :param query_delay: override the global query_delay.
       super().wait_for(query_delay or self.query_delay)
   def read(self):
       """Read from the device and check the response.
       Assert that the response starts with the device address.
       got = super().read()
       if got.startswith(self.address):
           return got[3:]
       else:
           raise ConnectionError(f"Expected message address '{self.address}', but read '
→{got[3:]}' for wrong address '{got[:3]}'.")
   voltage = Instrument.measurement(
        ":VOLT:?", """Measure the voltage in Volts.""")
```

If the device is initialized with address=12, a request for the voltage would send "012:VOLT:?" to the device and expect a response beginning with "012".

# 10.8.3 Bytes communication

Some devices do not expect ASCII strings but raw bytes. In those cases, you can call the write\_bytes() and read\_bytes() in your write() and read() methods. The following example shows an instrument, which has registers to be written and read via bytes sent.

```
class ExtremeBytes(Instrument):
    """Control the ExtremeBytes instrument with byte-based communication."""
    def __init__(self, adapter, name="ExtremeBytes"):
        super().__init__(adapter, name)

def write(self, command):
    """Write to the device according to the comma separated command.

    :param command: R or W for read or write, hexadecimal address, and data.

"""
function, address, data = command.split(",")
    b = [0x03] if function == "R" else [0x10]
```

```
b.extend(int(address, 16).to_bytes(2, byteorder="big"))
       b.extend(int(data).to_bytes(length=8, byteorder="big", signed=True))
       self.write_bytes(bytes(b))
   def read(self):
       """Read the response and return the data as a string, if applicable."""
       response = self.read_bytes(2) # return type and payload
       if response [0] == 0x00:
           raise ConnectionError(f"Device error of type {response[1]} occurred.")
       if response [0] == 0x03:
           # read that many bytes and return them as an integer
           data = self.read_bytes(response[1])
           return str(int.from_bytes(data, byteorder="big", signed=True))
       if response[0] == 0x10 and response[1] != 0x00:
           raise ConnectionError(f"Writing to the device failed with error {response[1]}
")
   voltage = Instrument.control(
       "R, 0x106,1", "W, 0x106,%i",
       """Control the output voltage in mV.""",
```

# 10.9 Writing tests

Tests are very useful for writing good code. We have a number of tests checking the correctness of the pymeasure implementation. Those tests (located in the tests directory) are run automatically on our CI server, but you can also run them locally using pytest.

When adding instruments, your primary concern will be tests for the *instrument driver* you implement. We distinguish two groups of tests for instruments: the first group does not rely on a connected instrument. These tests verify that the implemented instrument driver exchanges the correct messages with a device (for example according to a device manual). We call those "protocol tests". The second group tests the code with a device connected.

Implement device tests by adding files in the tests/instruments/... directory tree, mirroring the structure of the instrument implementations. There are other instrument tests already available that can serve as inspiration.

#### 10.9.1 Protocol tests

In order to verify the expected working of the device code, it is good to test every part of the written code. The <code>expected\_protocol()</code> context manager (using a <code>ProtocolAdapter</code> internally) simulates the communication with a device and verifies that the sent/received messages triggered by the code inside the with statement match the expectation.

```
import pytest
from pymeasure.test import expected_protocol
from pymeasure.instruments.extreme5000 import Extreme5000
def test_voltage():
```

```
"""Verify the communication of the voltage getter."""
with expected_protocol(
    Extreme5000,
    [(":VOLT 0.345", None),
        (":VOLT?", "0.3000")],
) as inst:
    inst.voltage = 0.345
    assert inst.voltage == 0.3
```

In the above example, the imports import the pytest package, the expected\_protocol and the instrument class to be tested.

The first parameter, Extreme5000, is the class to be tested.

When setting the voltage, the driver sends a message (":VOLT 0.345"), but does not expect a response (None). Getting the voltage sends a query (":VOLT?") and expects a string response ("0.3000"). Therefore, we expect two pairs of send/receive exchange. The list of those pairs is the second argument, the expected message protocol.

The context manager returns an instance of the class (inst), which is then used to trigger the behaviour corresponding to the message protocol (e.g. by using its properties).

If the communication of the driver does not correspond to the expected messages, an Exception is raised.



The expected messages are **without** the termination characters, as they depend on the connection type and are handled by the normal adapter (e.g. VISAAdapter).

Some protocol tests in the test suite can serve as examples:

- Testing a simple instrument: tests/instruments/keithley/test\_keithley2000.py
- Testing a multi-channel instrument: tests/instruments/tektronix/test\_afg3152.py
- Testing instruments using frame-based communication: tests/instruments/hcp/tc038.py

#### **Test generator**

In order to facilitate writing tests, if you already have working code and a device at hand, we have a *Generator* for tests. You can control your instrument with the TestGenerator as a middle man. It logs the method calls, the device communication and the return values, if any, and writes tests according to these log entries.

```
from pymeasure.generator import Generator
from pymeasure.instruments.hcp import TC038

generator = Generator()
inst = generator.instantiate(TC038, adapter, 'hcp', adapter_kwargs={'baud_rate': 9600})
```

As a first step, this code imports the Generator and generates a middle man instrument. The instantiate() method creates an instrument instance and logs the communication at startup. The Generator creates a special adapter for the communication with the device. It cannot inspect the instrument's \_\_init\_\_(), however. Therefore you have to specify the all connection settings via the adapter\_kwargs dictionary, even those, which are defined in \_\_init\_\_(). These adapter arguments are not written to tests. If you have arguments for the instrument itself, e.g. a RS485 address, you may give it as a keyword argument. These additional keyword arguments are included in the tests.

10.9. Writing tests 677

Now we can use inst as if it were created the normal way, i.e. inst = TC038(adapter), where adapter is some resource string. Having gotten and set some properties, and called some methods, we can write the tests to a file.

The following data will be written to file:

```
import pytest
from pymeasure.test import expected_protocol
from pymeasure.instruments.hcp import TC038
def test_init():
   with expected_protocol(
            TC038,
            [(b'\x0201010WRS01D0002\x03', b'\x0201010K\x03')],
   ):
        pass # Verify the expected communication.
def test_information_getter():
   with expected_protocol(
            [(b'\x0201010WRS01D0002\x03', b'\x0201010K\x03'),
             (b'\x0201010INF6\x03', b'\x0201010KUT150333 \V01.R001111222233334444\x03')],
   ) as inst:
        assert inst.information == 'UT150333 V01.R001111222233334444'
@pytest.mark.parametrize("comm_pairs, value", (
    ([(b'\x0201010WRS01D0002\x03', b'\x0201010K\x03'),
      (b'\x0201010WWRD0120,01,00C8\x03', b'\x0201010K\x03')],
     20),
    ([(b'\x0201010WRS01D0002\x03', b'\x0201010K\x03'),
      (b'\x0201010WWRD0120,01,0258\x03', b'\x0201010K\x03')],
     60),
def test_setpoint_setter(comm_pairs, value):
   with expected_protocol(
            TC038,
            comm_pairs,
   ) as inst:
        inst.setpoint = value
def test_setpoint_getter():
   with expected_protocol(
```

(continued from previous page)

```
TC038,
        [(b'\x0201010WRS01D0002\x03', b'\x0201010K\x03'),
        (b'\x0201010WRDD0120,01\x03', b'\x0201010K00C8\x03')],
) as inst:
    assert inst.setpoint == 20.0
```

#### 10.9.2 Device tests

It can be useful as well to test the code against an actual device. The necessary device setup instructions (for example: connect a probe to the test output) should be written in the header of the test file or test methods. There should be the connection configuration (for example serial port), too. In order to distinguish the test module from protocol tests, the filename should be test\_instrumentName\_with\_device.py, if the device is called instrumentName.

To make it easier for others to run these tests using their own instruments, we recommend to use pytest. fixture to create an instance of the instrument class. It is important to use the specific argument name connected\_device\_address and define the scope of the fixture to only establish a single connection to the device. This ensures two things: First, it makes it possible to specify the address of the device to be used for the test using the --device-address command line argument. Second, tests using this fixture, i.e. tests that rely on a device to be connected to the computer are skipped by default when running pytest. This is done to avoid that tests that require a device are run when none is connected. It is important that all tests that require a connection to a device either use the connected\_device\_address fixture or a fixture derived from it as an argument.

A simple example of a fixture that returns a connected instrument instance looks like this:

```
@pytest.fixture(scope="module")
def extreme5000(connected_device_address):
   instr = Extreme5000(connected_device_address)
   # ensure the device is in a defined state, e.g. by resetting it.
   return instr
```

Note that this fixture uses connected\_device\_address as an input argument and will thus be skipped by automatic test runs. This fixture can then be used in a test functions like this:

```
def test_voltage(extreme5000):
    extreme5000.voltage = 0.345
    assert extreme5000.voltage == 0.3
```

Again, by specifying the fixture's name, in this case extreme5000, invoking pytest will skip these tests by default.

It is also possible to define derived fixtures, for example to put the device into a specific state. Such a fixture would look like this:

```
@pytest.fixture
def auto_scaled_extreme5000(extreme5000):
    extreme5000.auto_scale()
    return extreme5000
```

In this case, do not specify the fixture's scope, so it is called again for every test function using it.

To run the test, specify the address of the device to be used via the --device-address command line argument and limit pytest to the relevant tests. You can filter tests with the -k option or you can specify the filename. For example, if your tests are in a file called test\_extreme5000\_with\_device.py, invoke pytest with pytest -k extreme5000 --device-address TCPIP::192.168.0.123::INSTR".

10.9. Writing tests 679

There might also be tests where manual intervention is necessary. In this case, skip the test by prepending the test function with a @pytest.mark.skip(reason="A human needs to press a button.") decorator.

## 10.10 Solutions for implementation challenges

This is a list of less common challenges, their solutions, and example instruments.

### 10.10.1 General issues

• Small numbers (<1e-5) are shown as 0 with %f. If an instrument understands exponential notation, you can use %g, which switches between floating point and exponential format, depending on the exponent.

## 10.10.2 Communication protocol issues

- Binary, frame-based communication, see hcp. TC038D
- All replies have the same length, see *aja.DCXS*
- The device generates garbage messages at startup, cluttering the buffer, see a ja. DCXS
- An instrument and its channel need to override values, but it has to use the correct ask method as well, see tcpowerconversion.CXN

#### 10.10.3 Channels

- Not all channels have the same features, see MKS937B
- Channel names in the communication (1, 2, 3) differ from front panel (A, B, C), see AdvantestR624X
- A family of instruments in which a property of the channels is different for different members of the family, see
   AnritsuMS464xB
- If you want to document the type of your instrument's channels (with a clickable link), check out the source of the *Aim-TTI PL Series Power Supplies* page.

**CHAPTER** 

**ELEVEN** 

## **CODING STANDARDS**

In order to maintain consistency across the different instruments in the PyMeasure repository, we enforce the following standards.

# 11.1 Python style guides

The PEP8 style guide and PEP257 docstring conventions should be followed.

Function and variable names should be lower case with underscores as needed to separate words. CamelCase should only be used for class names, unless working with Qt, where its use is common.

In addition, there is a configuration for the flake8 linter present. Our codebase should not trigger any warnings. Many editors/IDEs can run this tool in the background while you work, showing results inline. Alternatively, you can run flake8 in the repository root to check for problems. In addition, our automation on Github also runs some checkers. As this results in a much slower feedback loop for you, it's not recommended to rely only on this.

It is allowed but not required to use the black code formatter. To avoid introducing unrelated changes when working on an existing file, it is recommended to use the darker tool instead of *black*. This helps to keep the focus on the implementation instead of unrelated formatting, and thereby facilitates code reviews. darker is compatible with black, but only formats regions that show as changed in Git. If there are conflicts between black/darker's output and flake8 (especially related to E203), flake8 takes precedence. Use #noqa: E203 to disable E203 warnings for a specific line if appropriate.

You may add type hints as you see fit. All type hints should adhere to the guidelines set out in the typing package.

### 11.2 Documentation

PyMeasure documents code using reStructuredText and the Sphinx documentation generator. All functions, classes, and methods should be documented in the code using a docstring, see section *Docstrings*.

# 11.3 Usage of getter and setter functions

Getter and setter functions are discouraged, since properties provide a more fluid experience. Given the extensive tools available for defining properties, detailed in the sections starting with *Writing properties*, these types of properties are preferred.

# 11.4 Docstrings

Descriptive and specific docstrings for your properties and methods are important for your users to quickly glean important information about a property. It is advisable to follow the PEP257 docstring guidelines. Most importantly:

- Use triple-quoted strings (""") to delimit docstrings.
- One short summary line in imperative voice, with a period at the end.
- Optionally, after a blank line, include more detailed information.
- For functions and methods, you can add documentation on their parameters using the reStructuredText docstring format

Specific to properties, start them with "Control", "Get", "Measure", or "Set" to indicate the kind of property, as it is not visible after import, whether a property is gettable ("Get" or "Measure", e.g. for a measurement()), settable ("Set", for a setting()), or both ("Control", for a control()). In addition, it is useful to add type and information about *Restricting values with validators* (if applicable) at the end of the summary line, see the docstrings shown in examples throughout the *Adding instruments* section. For example a docstring could be """Control the voltage in Volts (float strictly from -1 to 1).""".

The docstring is for information that is relevant for *using* a property/method. Therefore, do *not* add information about internal/hidden details, like the format of commands exchanged with the device.

## **TWELVE**

## **AUTHORS**

PyMeasure was started in 2013 by Colin Jermain and Graham Rowlands at Cornell University, when it became apparent that both were working on similar Python packages for scientific measurements. PyMeasure combined these efforts and continues to gain valuable contributions from other scientists who are interested in advancing measurement software.

The following developers have contributed to the PyMeasure package:

Colin Jermain

Graham Rowlands

Minh-Hai Nguyen

Guen Prawiro-Atmodjo

Tim van Boxtel

Davide Spirito

Marcos Guimaraes

Ghislain Antony Vaillant

Ben Feinstein

Neal Reynolds

Christoph Buchner

Julian Dlugosch

Sylvain Karlen

Joseph Mittelstaedt

Troy Fox

Vikram Sekar

Casper Schippers

Sumatran Tiger

Michael Schneider

Dennis Feng

Stefano Pirotta

Moritz Jung

Richard Schlitz

Manuel Zahn

Mikhaël Myara

Domenic Prete

Mathieu Jeannin

Paul Goulain

John McMaster

Dominik Kriegner

Jonathan Larochelle

Dominic Caron

Mathieu Plante

Michele Sardo

Steven Siegl

(continues on next page)

(continued from previous page)

Benjamin Klebel-Knobloch

Markus Röleke

Demetra Adrahtas

Dan McDonald

Hud Wahab

Nicola Corna

Robert Eckelmann

Sam Condon

Andreas Maeder

Bastian Leykauf

Matthew Delaney

Marco von Rosenberg

Jack Van Sambeek

JC Arbelbide

Florian Jünger

Benedikt Moneke

Asaf Yagoda

Fabio Garzetti

Daniel Schmeer

Mike Manno

David Sanchez Sanchez

Andres Ruz-Nieto

Carlos Martinez

Scott Candev

Tom Verbeure

Juraj Jašík

Max Herbold

Alexander Wichers

Ashok Bruno

Robert Roos

Sebastien Weber

Sebastian Neusch

Felix Thouin

Ulrich Sauter

Guus Kuiper

Bernhard Lang

Armindo Pinto

Frank Wu

Heinz-Alexander Fütterer

Per-Olof Svensson

Karl Komierowski

Alec Vercruysse

Matthew Zenaldin

Canyon Clark

Connor Carr

Jannis Kleine-Schönepauck

Douwe den Blanken

Till Zürner

J. A. Wilcox

Nick James Kirkby

Konrad Gralher

David Ziliak

(continues on next page)

(continued from previous page)

David Sun Kévin Petit

Félix Thouin

Orion Smedley

Samuel Simpson

Isha Sharma

Dawid Maślanka

Felix Thouin

Emmanuel Ferdman

Johannes Rothmayr

Fred Gillard

Dimitrios Simatos

Dongkai Pan

Julian van Doorn

Jörg Wunsch

Falk Korndörfer

Henk Vergone

Sergey Ilyev

Alfons Schuck

Patrick Mischke

686 Chapter 12. Authors

**CHAPTER** 

## **THIRTEEN**

## **LICENSE**

### Copyright (c) 2013-2025 PyMeasure Developers

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

688 Chapter 13. License

**CHAPTER** 

## **FOURTEEN**

## **CHANGELOG**

# 14.1 Upcoming Release

### **14.1.1 Removed**

- Remove support for Python 3.8
- Remove deprecated Adapter and Instrument parameters preprocess\_reply, query\_delay
- Remove deprecated Adapter methods ask, values, and binary\_values
- · Remove deprecated Telnet and VXI adapters

## 14.1.2 Changed features

• Instrument.control does not apply get\_process to a returned list anymore, only to a single value. Use get\_process\_list parameter instead for processing a list of values.

### 14.1.3 Deprecated

- Replaced sensitvity attribute of pymeasure/instruments/srs/SR860.py by sensitivity
- Replaced filer\_synchronous attribute of pymeasure/instruments/srs/SR860.py by filter\_synchronous

# 14.2 Version 0.15.0 (2025-01-15)

Main items of this new release:

- pyproject.toml replaces setup.cfg
- 12 new instruments

### 14.2.1 Deprecated features

• The FSL class imported from pymeasure/instruments/rohdeschwarz/fsl.py is deprecated, the current version can be found in pymeasure/instruments/rohdeschwarz/fsseries.py.

#### 14.2.2 Instruments

- Add Agilent E5062A (@AlecVercruysse, #1146)
- Add Hewlett Packard 8753E VNA (@Sionwage, #1004)
- Add Keithley DAQ6510 (@Aphelion82, #1128)
- Add Keithley2281S (@PfannenHans, #1054)
- Add LD400P electronic load (@RobertoRoos, #1168)
- Add Philips PM6669 Universal Frequency Counter (@dirkjankrijnders, #570)
- Add Rohde&Schwarz FSSeries class for instruments such as FSL (previously stand-alone class), FSW and others
   (@jnnskls, #1156) Previously, the instrument class FSL could be imported from pymeasure/instruments/
   rohdeschwarz/fsl.py, which can now be imported from pymeasure/instruments/rohdeschwarz/
   fsseries.py alongside the base class FSSeries and FSW.
- Add Rigol dg822 waveform generator (@fthouin, #1159)
- Add Siglent SDS1072CML oscilloscope (@fthouin, #1080, 1195)
- Add more Thyracont vacuum transducer device types (@dkriegner, #1143)
- Fix Agilent E5062A: docs + rename active\_traces to visible\_traces (@AlecVercruysse, #1174)
- Fix AnritsuMS2090A: Confirm SCPIMixin (@icchalmers, #1191)
- Fix race condition in test suite due to HP8116A tests (@kpet, #1145)
- Fix description of Keithley2000.measure\_resistance method (@J3NZ0L, #1193)
- Fix test for Keithley6510 (@BenediktBurger, #1135)
- Fix Kepco BOP power supplies documentation (@JAW90, #1199)
- Update AFG3152CChannel to docs (@ssimpson-ens, #1176)
- Update Keysight E3631A with added output\_enabled property to individual channels (@inonRAAAM, #1209)
- Update Teledyne with VBS methods (@RobertoRoos, #1166)

### 14.2.3 Automation

- Explicitly set encoding to utf8 when writing and reading data to file, allowing the use of special characters. Previously the encoding was not explicitly set, this could potentially disrupt loading old data-files; if this is required, the encoding can be changed by changing (e.g., monkey-patching) the pymeasure.experiment. Results.ENCODING property. (@CasperSchippers, #1123)
- Add has\_next method to Procedure class (@Did-Mas, #1185)
- Change logger from root to module-specific logger. (@mmerlo, #1165)

### 14.2.4 Documentation

- Fix a couple of typos in procedure.rst (@kpet, #1141)
- Update contribute.rst also for GitHub Desktop (@bernhardlang, @OrionSmedley, #983, #1171)
- Fix documentation errors (@msmttchr, @kpet, #1133, #1148)
- Update InstrumentKit reference (@emmanuel-ferdman, #1187)
- Change copyright year to 2025 (@BenediktBurger, #1210)

#### 14.2.5 Miscellaneous

- Prepare for v0.14.0 (@BenediktBurger, #1104)
- add "Reaveal in File Explorer" to browser item context menu (@Did-Mas, #1188)
- Update numpy 2 test (@CasperSchippers, #1126)
- Added an install group for building documentation (@RobertoRoos, #1180)
- Removed setup.cfg, replaced by pyproject.toml (@RobertoRoos, #1182)

#### 14.2.6 New Contributors

@Sionwage, @bernhardlang, @Aphelion82, @kpet, @mmerlo, @OrionSmedley, @ssimpson-ens, @Did-Mas, @emmanuel-ferdman, @icchalmers, @fthouin, @PfannenHans, @dirkjankrijnders, @J3NZ0L, @inonRAAAM,

**Full Changelog:** https://github.com/pymeasure/pymeasure/compare/v0.14.0...v0.15.0

# 14.3 Version 0.14.0 (2024-05-22)

Main items of this new release:

- Add support for numpy 2.0
- Add support for python 3.12
- Improve academic quotability with an up to date Zenodo DOI and with citation information.
- Add default queue method and a FileInputWidget, allowing to more quickly get started with the PyMeasure user interface (ManagedWindow).
- Add a SCPIMixin base class for instruments instead of defining includeSCPI=True
- Instrument manufacturer modules are no longer imported in the pymeasure/instruments/\_\_init\_\_.py file. Previously, when importing a single instrument into a procedure, all instruments would be imported into memory through the manufacturer modules in pymeasure/instruments/\_\_init\_\_.py. Removing manufacturer modules from that file lowers the memory footprint of pymeasure when importing an instrument. Instrument classes will need to be imported from the manufacturer module or explicitly from the instrument driver file. For example, from pymeasure.instruments import Extreme5000 will need to change to from pymeasure.instruments.extreme import Extreme5000 or from pymeasure.instruments. extreme.extreme5000 import Extreme5000.
- 17 new instruments

## 14.3.1 Deprecated features

- Remove TelnetAdapter, as its library is deprecated (@BenediktBurger, #1045)
- Replaced directory\_input keyword-argument of ManagedWindowBase by enable\_file\_input (@Casper-Schippers, #964)
- Make parameter includeSCPI obligatory for all instruments, even those which use SCPI (@BenediktBurger, #1007)
- Setting includeSCPI=True is deprecated, inherit instead the SCPIMixin class if the device supports SCPI commands.
- Replaced celcius attribute of LakeShoreTemperatureChannel by celsius (@afuetterer, #1003)
- Replaced error property of Keithley instruments by next\_error.
- Replaced measurement\_time property of Pendulum CNT-91 by gate\_time.
- Replaced sample\_rate keyword-argument of buffer\_frequency\_time\_series of Pendulum CNT-91 by gate\_time.
- Replaced MKS937B unit to use instruments/mksinst/mks937b/Unit instead of strings (@dkriegner, @BenediktBurger #1034)

### 14.3.2 Instruments mechanics

- Add a SCPI base class SCPIMixin as replacement for includeSCPI=True (@BenediktBurger, #905, #1007, #1019, #1047)
- Add next\_error property to SCPI instruments (@BenediktBurger, #1024)
- Make query\_delay=None the default for wait\_for (@BenediktBurger, #1077)
- Fix expected\_protocol using empty dictionary as default value (@BenediktBurger, #1087)
- Remove auto-importing all instruments in pymeasure/instruments/\_\_init\_\_.py` (@mcdo0486, #919)
- Add find\_serial\_port to find a serial port by providing USB information (@BenediktBurger, #982)

#### 14.3.3 Instruments

- Add Agilent4294A (@driftregion, #998)
- Add Agilent 4284A by (@ConnorGCarr #1079)
- Add AimTTI PL series power supplies (@guuskuiper, #942)
- Add HP11713A Switch & Attenuator Driver (@neuschs, #970)
- Add HP437B power meter (@neuschs, #979)
- Add Inficon SQM160 SQM-160 multi-film rate/thickness monitor (@dkriegner, #991)
- Add Keithley 2182 (@ConnorGCarr, #1043)
- Add KeithleyDMM6500 (@fwutw, #963)
- Add Kepco BOP 36-12 Bipolar Power Supply (@JAW90, #1086)
- Add KeysightE3631A (@OptimisticBeliever, #990)
- Add Kuhne Electronic KU SG 2.45 250A microwave generator (@jurajjasik, @BenediktBurger, @1108)

- Add MKS 974B vacuum pressure transducer (@dkriegner, #1034)
- Add Proterial rod4 (@ConnorGCarr, #1044)
- Add Racal-Dana 1992 universal counter (@tomverbeure, #798, #1012)
- Add redpitaya board (@seb5g, #1010, #1035)
- Add Teledyne HDO6xxx (@RobertoRoos, #868)
- Add Yokogawa AQ6370D Optical Spectral Analyzer (@jnnskls, #1059)
- Fix property docstrings of several instruments (@BenediktBurger, #1018)
- Fix checksums of hcp TC038D tests (@BenediktBurger, #987)
- Fix Hp8116a (@BenediktBurger, #1088)
- Fix Hp856x to append amplitude units (@neuschs, #977)
- Fix Keysight E36312A confirmed SCPI functionality (@Konradrundfunk, #1107)
- Fix Stanford Research SR830 output conversion (@dkriegner, #1069)
- Fix SR830 missing get\_buffer method (@seb5g, #999)
- Fix set command of SR860 aux output (@wehlgrundspitze, #1048)
- Fix Temptronic test to use ns perf counter (@BenediktBurger, #1109, #1110)
- Fix Toptica Ibeamsmart referencing removed adapter function (@BenediktBurger, #1065)
- Fix typos in docstrings for Keithley instruments (@V0XNIHILI, #1071)
- Link Keysight, Agilent, and HP documentation pages. (@BenediktBurger, #1021)
- Update Agilent33500 Series from .ch[] to .channels[] (@AlecVercruysse, #945)
- Update AWG401x driver to use 'channels' (@mcdo0486, #944)
- Update HP33120A with new burst modulation parameters (@mzen228, #1056)
- Update HP34401A with new remote control command. (@Rybok, #992)
- Update Keithleys' next\_error (@msmttchr, #1030)
- Update pendulum CNT-91 (@bleykauf, #988)

### 14.3.4 GUI

- Add a FileInputWidget to choose if and where the experiment data is stored. (@CasperSchippers, #964)
- Add a default Queue method for ManagedWindowBase is implemented. (@CasperSchippers, #964)
- Fix ScientificInput to be locale compatible (@pyZerrenner, #1074)
- Fix exception if loading result file with an empty parameter (@poje42, #1016)

### 14.3.5 Miscellaneous

- Add support for python 3.12 (@BenediktBurger, #1051)
- Add support for numpy 2.0 (@CasperSchippers, #1026)
- Add codecov to CI and to readme (@BenediktBurger, #1037, #1052, #1099)
- Add citation file for PyMeasure repository (@mcdo0486, #1092)
- Add release CI (@BenediktBurger, #1039)
- Update readme with permanent Zenodo DOI (@BenediktBurger, #1095)
- Bump CI dependencies to: pyvisa 1.13.0, checkout@v4 (@mcdo0486, #1097)
- Fix/pandas futurewarning (@CasperSchippers, #1062)
- Change copyright year. (@BenediktBurger, #1032)
- Fix typos (@afuetterer, #1003)

### 14.3.6 New Contributors

@guuskuiper, @OptimisticBeliever, @fwutw, @afuetterer, @poje42, @Rybok, @AlecVercruysse, @ConnorGCarr, @mzen228, @jnnskls, @V0XNIHILI, @pyZerrenner, @JAW90, @driftregion, @jurajjasik, @Konradrundfunk

**Full Changelog:** https://github.com/pymeasure/pymeasure/compare/v0.13.1...v0.14.0

## 14.4 Version 0.13.1 (2023-10-05)

New release to fix ineffective python version restriction in the project metadata (only affected Python<=3.7 environments installing via pip).

# 14.5 Version 0.13.0 (2023-09-23)

Main items of this new release:

- Dropped support for Python 3.7, added support for Python 3.11.
- Adds a test generator, which observes the communication with an actual device and writes protocol tests accordingly.
- 2 new instrument drivers have been added.

## 14.5.1 Deprecated features

• Attocube ANC300: The stepu and stepd properties are deprecated, use the new move\_raw method instead. (@dkriegner, #938)

### 14.5.2 Instruments

- Adds a test generator (@bmoneke, #882)
- Adds Thyracont Smartline v2 vacuum sensor transmitter (@bmoneke, #940)
- Adds Thyracont Smartline v1 vacuum gauge (@dkriegner, #937)
- AddsTeledyne base classes with most of LeCroyT3DSO1204 functionality (@RobertoRoos, #951)
- Fixes instrument documentation (@mcdo0486, #941, #903, @omahs, #960)
- Fixes Toptica Ibeamsmart's \_\_init\_\_ (@waveman68, #959)
- Fixes VISAAdapter flush\_read\_buffer() (@ileu, #968)
- Updates Keithley2306 and AFG3152C to Channels (@bilderbuchi, #953)

### 14.5.3 GUI

- Adds console mode (@msmttchr, #500)
- Fixes Dock widget (@msmttchr, #961)

#### 14.5.4 Miscellaneous

- Change CI from conda to mamba (@bmoneke, #947)
- Add support for python 3.11 (@CasperSchippers, #896)

### 14.5.5 New Contributors

@waveman68, @omahs, @ileu

Full Changelog: https://github.com/pymeasure/pymeasure/compare/v0.12.0...v0.13.0

# 14.6 Version 0.12.0 (2023-07-05)

Main items of this new release:

- A Channel base class has been added for easier implementation of instruments with channels.
- 19 new instrument drivers have been added.
- Added tests for some commonalities across all instruments.
- We continue to clean up our API in preparation for a future version 1.0. Deprecations and subsequent removals are listed below.

## 14.6.1 Deprecated features

- HP 34401A: voltage\_ac, current\_dc, current\_ac, resistance, resistance\_4w properties, use function\_ and reading properties instead.
- Toptica IBeamSmart: channel1\_enabled, use ch\_1.enabled property instead (equivalent for channel2). Also laser\_enabled is deprecated in favor of emission (@bmoneke, #819).
- TelnetAdapter: use VISAAdapter instead. VISA supports TCPIP connections. Use the resource\_name TCPIP[board]::<hostname>::connect to a server (@Max-Herbold, #835).
- Attocube ANC300: host argument, pass a resource string or adapter as Adapter passed to Instrument. Now
  communicates through the VISAAdapter rather than deprecated TelnetAdapter. The initializer now accepts
  name as its second keyword argument so all previous initialization positional arguments (axisnames, passwd,
  query delay) should be switched to keyword arguments.
- The property creators control, measurement, and setting do not accept arbitrary keyword arguments anymore. Use the v\_kwargs parameter for arguments you want to pass on to values method, instead.
- The property creators control, measurement, and setting do not accept *command\_process* anymore. Use a dynamic property or a *Channel* instead, as appropriate (@bmoneke, #878).
- See also the next section.

## 14.6.2 New adapter and instrument mechanics

- All instrument constructors are required to accept a name argument.
- Changed: read\_bytes of all Adapters by default does not stop reading on a termination character, unless the new argument break\_on\_termchar is set to *True*.
- Channel class added. Instrument.channels and Instrument.ch\_X (X is any channel name) are reserved attributes for channel mechanics.
- The parameters check\_get\_errors and check\_set\_errors enable calling methods of the same name. This enables more systematically dealing with instruments that acknowledge every "set" command.
- Adds Channel feature to instruments (@bmoneke, mcdo0486, #718, #761, #852, #931)
- Adds maxsplit parameter to values method (@bmoneke, #793)
- Adds (deprecated) global preprocess reply for backward compatibility (@bmoneke, #876)
- Adds fallback version for discarding the read buffer to VISAAdapter (@dkriegner, #836)
- Adds flush\_read\_buffer to SerialAdapter (@RobertoRoos, #865)
- Adds gpib\_read\_timeout to PrologixAdapter (@neuschs, #927)
- Adds command line option to pass resource address for instrument tests (@bleykauf, #789)
- Adds "find all instruments" and channels for testing (@bmoneke, #909, @mcdo0486, #911, #912)
- Adds test that an instrument hands kwargs to the adapter (@bmoneke, #814)
- Adds property docstring check (@bmoneke, #895)
- Improves property factories' docstrings (@bmoneke, #843)
- Improves property factories: do not allow undefined kwargs (@bmoneke, #856)
- Improves property factories: check\_set/get\_errors argument to call methods of the same name (@bmoneke, #883)

- Improves read\_bytes of Adapter (@bmoneke, #839)
- Improves the Protocol Adapter with a mock connection (@bmoneke, #782), and enable it to have empty messages in the protocol (@bmoneke, #818)
- Improves Prologix adapter documentation (@bmoneke, #813) and configurable settings (@bmoneke, #845)
- Improves behavior of read\_bytes(-1) for SerialAdapter (@RobertoRoos, #866)
- Improves all instruments with name kwarg (@bmoneke, #877)
- Improves VisaAdapter: close manager only when using pyvisa-sim (@dkriegner, #900)
- Harmonises instrument name definition pattern, consistently name the instrument connection argument "adapter"
   (@bmoneke, #659)
- Fixes ProtocolAdapter has list in signature (@bmoneke, #901)
- Fixes VISAAdapter's read\_bytes (@bmoneke, #867)
- Fixes query\_delay usage in VISAAdapter (@bmoneke, #765)
- Fixes VisaAdapter: close resource manager only when using pyvisa-sim (@dkriegner, #900)

### 14.6.3 Instruments

- New Advantest R624X DC Voltage/Current Sources/Monitors (@wichers, #802)
- New AJA International DC sputtering power supply (@dkriegner, #778)
- New Anritus MS2090A (@aruznieto, #787)
- New Anritsu MS4644B (@CasperSchippers, #827)
- New DSP 7225 and new DSPBase instrument (@mcdo0486, #902)
- New HP 8560A / 8561B Spectrum Analyzer (@neuschs, #888)
- New IPG Photonics YAR Amplifier series (@bmoneke, #851)
- New Keysight E36312A power supply (@scandey, #785)
- New Keithley 2200 power supply (@ashokbruno, #806)
- New Lake Shore 211 Temperature Monitor (@mcdo0486, #889)
- New Lake Shore 224 and improves Lakeshore instruments (@samcondon4, #870)
- New MKS Instruments 937B vacuum gauge controller (@dkriegner, @bilderbuchi, #637, #772, #936)
- New Novanta FPU60 laser power supply unit (@bmoneke, #885)
- New TDK Lambda Genesys 80-65 DC and 40-38 DC power supplies (@mcdo0486, 906)
- New Teledyne T3AFG waveform generator instrument (@scandey, #791)
- New Teledyne (LeCroy) T3DSO1204 Oscilloscope (@LastStartDust, #697, @bilderbuchi, #770)
- New T&C Power Conversion RF power supply (@dkriegner, #800)
- New Velleman K8090 relay device (@RobertoRoos, #859)
- Improves Agilent 33500 with the new channel feature (@JCarl-OS, #763, #773)
- Improves HP 3478A with calibration data related functions (@tomverbeure, #777)
- Improves HP 34401A (@CodingMarco, #810)
- Improves the Oxford instruments with the new channel feature (@bmoneke, #844)

- Improves Siglent SPDxxxxX with the new channel feature (@AidenDawn 758)
- Improves Teledyne T3DSO1204 device tests (@LastStarDust, #841)
- Fixes Ametek DSP 7270 lockin amplifier issues (@seb5g, #897)
- Fixes DSP 7265 erroneously using preprocess\_reply (@mcdo0486, #873)
- Fixes print statement in DSPBase.sensitivity (@mcdo0486, #915)
- Fixes Fluke bath commands (@bmoneke, #874)
- Fixes a frequency limitation in HP 8657B (@LongnoseRob, #769)
- Fixes Keithley 2600 channel calling parent's shutdown (@mcdo0486, #795)

### 14.6.4 Automation

- Adds tolerance for opening result files with missing parameters (@msmttchr, #780)
- Validate DATA\_COLUMNS entries earlier, avoid exceptions in a running procedure (@mcdo0486, #796, #934)

### 14.6.5 GUI

- Adds docking windows (@mcdo0486, #722, #762)
- Adds save plot settings in addition to dock layout (@mcdo0486, #850)
- Adds log widget colouring and format option (@CasperSchippers, #890)
- Adds table widget (@msmttchr, #771)
- New sequencer architecture: decouples it from the graphical tree, adapts it for further expansions (@msmttchr, #518)
- Moves coordinates label to the pyqtgraph PlotItem (@CasperSchippers, #822)
- Fixes crashing ImageWidget at new measurement (@CasperSchippers, #790)
- Fixes checkboxes not working for groups in inputs-widget (@CasperSchippers, #794)

### 14.6.6 Miscellaneous

- Adds a collection of solutions for instrument implementation challenges (@bmoneke, #853, #861)
- Updates Tutorials/Making\_a\_measurement/ example\_codes (@sansanda, #749)

### 14.6.7 New Contributors

@JCarl-OS, @aruznieto, @scandey, @tomverbeure, @wichers, @Max-Herbold, @RobertoRoos

**Full Changelog:** https://github.com/pymeasure/pymeasure/compare/v0.11.1...v0.12.0

# 14.7 Version 0.11.1 (2022-12-31)

## 14.7.1 Adapter and instrument mechanics

• Fix broken *PrologixAdapter.gpib*. Due to a bug in *VISAAdapter*, you could not get a second adapter with that connection (#765).

**Full Changelog:** https://github.com/pymeasure/pymeasure/compare/v0.11.0...v0.11.1

## 14.7.2 Dependency updates

• Required version of PyQtGraph is increased from pyqtgraph >= 0.9.10 to pyqtgraph >= 0.12 to support new PyMeasure display widgets.

### 14.7.3 GUI

- Added ManagedDockWindow to allow multiple dockable plots (@mcdo0486, @CasperSchippers, #722)
- Move coordinates label to the pyqtgraph PlotItem (@CasperSchippers, #822)
- New sequencer architecture (@msmttchr, @CasperSchippers, @mcdo0486, #518)
- Added "Save Dock Layout" functionality to DockWidget context menu. (@mcdo0486, #762)

## 14.8 Version 0.11.0 (2022-11-19)

Main items of this new release:

- 11 new instrument drivers have been added
- A method for testing instrument communication without hardware present has been added, see the documentation
- The separation between Instrument and Adapter has been improved to make future modifications easier. Adapters now focus on the hardware communication, and the communication *protocol* should be defined in the Instruments. Details in a section below.
- The GUI is now compatible with Qt6.
- We have started to clean up our API in preparation for a future version 1.0. There will be deprecations and subsequent removals, which will be prominently listed in the changelog.

### 14.8.1 Deprecated features

In preparation for a stable 1.0 release and a more consistent API, we have now started formally deprecating some features. You should get warnings if those features are used.

- Adapter methods ask, values, binary\_values, use Instrument methods of the same name instead.
- Adapter parameter preprocess\_reply, override Instrument.read instead.
- Adapter.query\_delay in favor of Instrument.wait\_for().
- Keithley 2260B: enabled property, use output\_enabled instead.

## 14.8.2 New adapter and instrument mechanics

- Nothing should have changed for users, this section is mainly interesting for instrument implementors.
- Documentation in 'Advanced communication protocols' in 'Adding instruments'.
- Adapter logs written and read messages.
- Particular adapters (VISAAdapter etc.) implement the actual communication.
- Instrument.control getter calls Instrument.values.
- Instrument.values calls Instrument.ask, which calls Instrument.write, wait\_for, and read.
- All protocol quirks of an instrument should be implemented overriding Instrument.write and read.
- Instrument.wait\_until\_read implements waiting between writing and reading.
- reading/writing binary values is in the Adapter class itself.
- PrologixAdapter is now based on VISAAdapter.
- SerialAdapter improved to be more similar to VISAAdapter: read/write use strings, read/write\_bytes bytes. Support for termination characters added.

#### 14.8.3 Instruments

- New Active Technologies AWG-401x (@garzetti, #649)
- New Eurotest hpp\_120\_256\_ieee (@sansanda, #701)
- New HC Photonics crystal ovens TC038, TC038D (@bmoneke, #621, #706)
- New HP 6632A/6633A/6634A power supplies (@LongnoseRob, #651)
- New HP 8657B RF signal generator (@LongnoseRob, #732)
- New Rohde&Schwarz HMP4040 power supply. (@bleykauf, #582)
- New Siglent SPDxxxxX series Power Supplies (@AidenDawn, #719)
- New Temptronic Thermostream devices (@mroeleke, #368)
- New TEXIO PSW-360L30 Power Supply (@LastStarDust, #698)
- New Thermostream ECO-560 (@AidenDawn, #679)
- New Thermotron 3800 Oven (@jcarbelbide, #606)
- Harmonize instruments' adapter argument (@bmoneke, #674)
- Harmonize usage of shutdown method (@LongnoseRob, #739)
- Rework Adapter structure (@bmoneke, #660)
- Add Protocol tests without hardware present (@bilderbuchi, #634, @bmoneke, #628, #635)
- Add Instruments and adapter protocol tests for adapter rework (@bmoneke, #665)
- Add SR830 sync filter and reference source trigger (@AsafYagoda, #630)
- Add Keithley6221 phase marker phase and line (@AsafYagoda, #629)
- Add missing docstrings to Keithley 2306 battery simulator (@AidenDawn, #720)
- Fix hcp instruments documentation (@bmoneke, #671)
- Fix HPLegacyInstrument initializer API (@bilderbuchi, #684)

- Fix Fwbell 5080 implementation (@mcdo0486, #714)
- Fix broken documentation example. (@bmoneke, #738)
- Fix typo in Keithley 2600 driver (@mcdo0486, #615)
- Remove dynamic use of docstring from ATS545 and make more generic (@AidenDawn, #685)

### 14.8.4 Automation

- Add storing unitful experiment results (@bmoneke, #642)
- Add storing conditions in file (@CasperSchippers, #503)

### 14.8.5 GUI

- Add compatibility with Qt 6 (@CasperSchippers, #688)
- Add spinbox functionality for IntegerParameter and FloatParameter (@jarvas24, #656)
- Add "delete data file" button to the browser\_item\_menu (@jarvas24, #654)
- Split windows.py into a folder with separate modules (@mcdo0486, #593)
- Remove dependency on matplotlib (@msmttchr, #622)
- Remove deprecated access to QtWidgets through QtGui (@maederan201, #695)

### 14.8.6 Miscellaneous

- Update and extend documentation (@bilderbuchi, #712, @bmoneke, #655)
- Add PEP517 compatibility & dynamically obtaining a version number (@bilderbuchi, #613)
- Add an example and documentation regarding using a foreign instrument (@bmoneke, #647)
- Add black configuration (@bleykauf, #683)
- Remove VISAAdapter.has\_supported\_version() as it is not needed anymore.

## 14.8.7 New Contributors

@jcarbelbide, @mroeleke, @bmoneke, @garzetti, @AsafYagoda, @AidenDawn, @LastStarDust, @sansanda

**Full Changelog:** https://github.com/pymeasure/pymeasure/compare/v0.10.0...v0.11.0

# 14.9 Version 0.10.0 (2022-04-09)

Main items of this new release:

- 23 new instrument drivers have been added
- New dynamic Instrument properties can change their parameters at runtime
- Communication settings can now be flexibly defined per protocol
- Python 3.10 support was added and Python 3.6 support was removed.

Many additions, improvements and have been merged

### 14.9.1 Instruments

- New Agilent B1500 Data Formats and Documentation (@moritzj29)
- New Anaheim Automation stepper motor controllers (@samcondon4)
- New Andeen Hagerling capacitance bridges (@dkriegner)
- New Anritsu MS9740A Optical Spectrum Analyzer (@md12g12)
- New BK Precision 9130B Instrument (@dennisfeng2)
- New Edwards nXDS (10i) Vacuum Pump (@hududed)
- New Fluke 7341 temperature bath instrument (@msmttchr)
- New Heidenhain ND287 Position Display Unit Driver (@samcondon4)
- New HP 3478A (@LongnoseRob)
- New HP 8116A 50 MHz Pulse/Function Generator (@CodingMarco)
- New Keithley 2260B DC Power Supply (@bklebel)
- New Keithley 2306 Dual Channel Battery/Charger Simulator (@mfikes)
- New Keithley 2600 SourceMeter series (@Daivesd)
- New Keysight N7776C Swept Laser Source (@maederan201)
- New Lakeshore 421 (@CasperSchippers)
- New Oxford IPS120-10 (@CasperSchippers)
- New Pendulum CNT-91 frequency counter (@bleykauf)
- New Rohde&Schwarz SFM TV test transmitter (@LongnoseRob)
- New Rohde&Schwarz FSL spectrum analyzer (@bleykauf)
- New SR570 current amplifier driver (@pyMatJ)
- New Stanford Research Systems SR510 instrument driver (@samcondon4)
- New Toptica Smart Laser diode (@dkriegner)
- New Yokogawa GS200 Instrument (@dennisfeng2)
- Add output low grounded property to Keithley 6221 (@CasperSchippers)
- Add shutdown function for Keithley 2260B (@bklebel)
- Add phase control for Agilent 33500 (@corna)
- Add assigning "ONCE" to auto\_zero to Keithley 2400 (@mfikes)
- Add line frequency controls to Keithley 2400 (@mfikes)
- Add LIA and ERR status byte read properties to the SRS Sr830 driver (@samcondon4)
- Add all commands to Oxford Intelligent Temperature Controller 503 (@CasperSchippers)
- Fix DSP 7265 lockin amplifier (@CasperSchippers)
- Fix bug in Keithley 6517B Electrometer (@CasperSchippers)
- Fix Keithley2000 deprecated call to visa.config (@bklebel)

- Fix bug in the Keithley 2700 (@CasperSchippers)
- Fix setting of sensor flags for Thorlabs PM100D (@bleykauf)
- Fix SCPI used for Keithley 2400 voltage NPLC (@mfikes)
- Fix missing return statements in Tektronix AFG3152C (@bleykauf)
- Fix DPSeriesMotorController bug (@samcondon4)
- Fix Keithley2600 error when retrieving error code (@bicarlsen)
- Fix Attocube ANC300 with new SCPI Instrument properties (@dkriegner)
- Fix bug in wait\_for\_trigger of Agilent33220A (neal-kepler)

### 14.9.2 GUI

- Add time-estimator widget (@CasperSchippers)
- Add management of progress bar (@msmttchr)
- Remove broken errorbar feature (@CasperSchippers)
- Change of pen width for pyqtgraph (@maederan201)
- Make linewidth changeable (@CasperSchippers)
- Generalise warning in plotter section (@CasperSchippers)
- Implement visibility groups in InputsWidgets (@CasperSchippers)
- Modify navigation of ManagedWindow directory widget (@jarvas24)
- Improve Placeholder logic (@CasperSchippers)
- Breakout widgets into separate modules (@mcdo0486)
- Fix setSizePolicy bug with PySide2 (@msmttchr)
- Fix managed window (@msmttchr)
- Fix ListParameter for numbers (@moritzj29)
- Fix incorrect columns on showing data (@CasperSchippers)
- Fix procedure property issue (@msmttchr)
- Fix pyside2 (@msmttchr)

### 14.9.3 Miscellaneous

- Improve SCPI property support (@msmttchr)
- Remove broken safeKeyword management (@msmttchr)
- Add dynamic property support (@msmttchr)
- Add flexible API for defining connection configuration (@bilderbuchi)
- Add write binary values() to SerialAdapter (@msmttchr)
- Change an outdated pyvisa ask() to query() (@LongnoseRob)
- Fix ZMQ bug (@bilderbuchi)
- Documentation for passing tuples to control property (@bklebel)

- Documentation bugfix (@CasperSchippers)
- Fixed broken links in documentation. (@samcondon4)
- Updated widget documentation (@mcdo0486)
- Fix typo SCIP->SCPI (@mfikes)
- Remove Python 3.6, add Python 3.10 testing (@bilderbuchi)
- Modernise the code base to use Python 3.7 features (@bilderbuchi)
- Added image data generation to Mock Instrument class (@samcondon4)
- Add autodoc warnings to the problem matcher (@bilderbuchi)
- Update CI & annotations (@bilderbuchi)
- Test workers (@mcdo0486)
- Change copyright date to 2022 (@LongnoseRob)
- Removed unused code (@msmttchr)

#### 14.9.4 New Contributors

@LongnoseRob, @neal, @hududed, @corna, @Daivesd, @samcondon4, @maederan201, @bleykauf, @mfikes, @bicarlsen, @md12g12, @CodingMarco, @jarvas24, @mcdo0486!

Full Changelog: https://github.com/pymeasure/pymeasure/compare/v0.9...v0.10.0

## 14.10 Version 0.9 - released 2/7/21

- PyMeasure is now officially at github.com/pymeasure/pymeasure
- Python 3.9 is now supported, Python 3.5 removed due to EOL
- Move to GitHub Actions from TravisCI and Appveyor for CI (@bilderbuchi)
- New additions to Oxford Instruments ITC 503 (@CasperSchippers)
- New Agilent 34450A and Keysight DSOX1102G instruments (@theMashUp, @jlarochelle)
- Improvements to NI VirtualBench (@moritzj29)
- New Agilent B1500 instrument (@moritzj29)
- New Keithley 6517B instrument (@wehlgrundspitze)
- Major improvements to PyVISA compatibility (@bilderbuchi, @msmttchr, @CasperSchippers, @cjermain)
- New Anapico APSIN12G instrument (@StePhanino)
- Improvements to Thorelabs Pro 8000 and SR830 (@Mike-HubGit)
- New SR860 instrument (@StevenSiegl, @bklebel)
- Fix to escape sequences (@tirkarthi)
- New directory input for ManagedWindow (@paulgoulain)
- New TelnetAdapter and Attocube ANC300 Piezo controller (@dkriegner)
- New Agilent 34450A (@theMashUp)

- New Razorbill RP100 strain cell controller (@pheowl)
- Fixes to precision and default value of ScientificInput and FloatParameter (@moritzj29)
- Fixes for Keithly 2400 and 2450 controls (@pyMatJ)
- Improvments to Inputs and open\_file\_externally (@msmttchr)
- Fixes to Agilent 8722ES (@alexmcnabb)
- Fixes to QThread cleanup (@neal-kepler, @msmttchr)
- Fixes to Keyboard interrupt, and parameters (@CasperSchippers)

## 14.11 Version 0.8 – released 3/29/19

- Python 3.8 is now supported
- New Measurement Sequencer allows for running over a large parameter space (@CasperSchippers)
- New image plotting feature for live image measurements (@jmittelstaedt)
- Improvements to VISA adapter (@moritzj29)
- Added Tektronix AFG 3000, Keithley 2750 (@StePhanino, @dennisfeng2)
- Documentation improvements (@mivade)
- Fix to ScientificInput for float strings (@moritzj29)
- New validator: strict\_discrete\_range (@moritzj29)
- · Improvements to Recorder thread joining
- Migrating the ReadtheDocs configuration to version 2
- National Instruments Virtual Bench initial support (@moritzj29)

## 14.12 Version 0.7 – released 8/4/19

- Dropped support for Python 3.4, adding support for Python 3.7
- Significant improvements to CI, dependencies, and conda environment (@bilderbuchi, @cjermain)
- Fix for PyQT issue in ResultsDialog (@CasperSchippers)
- Fix for wire validator in Keithley 2400 (@Fattotora)
- Addition of source\_enabled control for Keithley 2400 (@dennisfeng2)
- Time constant fix and input controls for SR830 (@dennisfeng2)
- Added Keithley 2450 and Agilent 33521A (@hlgirard, @Endever42)
- Proper escaping support in CSV headers (@feph)
- Minor updates (@dvase)

## 14.13 Version 0.6.1 - released 4/21/19

- Added Elektronica SM70-45D, Agilent 33220A, and Keysight N5767A instruments (@CasperSchippers, @sumatrae)
- Fixes for Prologix adapter and Keithley 2400 (@hlgirard, @ronan-sensome)
- Improved support for SRS SR830 (@CasperSchippers)

## 14.14 Version 0.6 – released 1/14/19

- New VXI11 Adapter for ethernet instruments (@chweiser)
- PyQt updates to 5.6.0
- Added SRS SG380, Ametek 7270, Agilent 4156, HP 34401A, Advantest R3767CG, and Oxford ITC503 instrustruments (@sylkar, @jmittelstaedt, @vik-s, @troylf, @CasperSchippers)
- Updates to Keithley 2000, Agilent 8257D, ESP 300, and Keithley 2400 instruments (@watersjason, @jmittel-staedt, @nup002)
- Various minor bug fixes (@thosou)

### 14.15 Version 0.5.1 - released 4/14/18

- · Minor versions of PyVISA are now properly handled
- Documentation improvements (@Laogeodritt and @ederag)
- Instruments now have set\_process capability (@bilderbuchi)
- Plotter now uses threads (@dvspirito)
- Display inputs and PlotItem improvements (@Laogeodritt)

### 14.16 Version 0.5 – released 10/18/17

- Threads are used by default, eliminating multiprocessing issues with spawn
- Enhanced unit tests for threading
- Sphinx Doctests are added to the documentation (@bilderbuchi)
- Improvements to documentation (@JuMaD)

## 14.17 Version 0.4.6 - released 8/12/17

- Reverted multiprocessing start method keyword arguments to fix Unix spawn issues (@ndr37)
- Fixes to regressions in Results writing (@feinsteinben)
- Fixes to TCP support using cloudpickle (@feinsteinben)
- · Restructing of unit test framework

## 14.18 Version 0.4.5 - released 7/4/17

- Recorder and Scribe now leverage QueueListener (@feinsteinben)
- PrologixAdapter and SerialAdapter now handle Serial objects as adapters (@feinsteinben)
- Optional TCP support now uses cloudpickle for serialization (@feinsteinben)
- Significant PEP8 review and bug fixes (@feinsteinben)
- Includes docs in the code distribution (@ghisvail)
- Continuous integration support for Python 3.6 (@feinsteinben)

## 14.19 Version 0.4.4 - released 6/4/17

- · Fix pip install for non-wheel builds
- Update to Agilent E4980 (@dvspirito)
- Minor fixes for docs, tests, and formatting (@ghisvail, @feinsteinben)

### 14.20 Version 0.4.3 – released 3/30/17

- Added Agilent E4980, AMI 430, Agilent 34410A, Thorlabs PM100, and Anritsu MS9710C instruments (@TvBMcMaster, @dvspirito, and @mhdg)
- Updates to PyVISA support (@minhhaiphys)
- Initial work on resource manager (@dvspirito)
- Fixes for Prologix adapter that allow read-write delays (@TvBMcMaster)
- Fixes for conda environment on continuous integration

# 14.21 Version 0.4.2 - released 8/23/16

- New instructions for installing with Anaconda and conda-forge package (thanks @melund!)
- Bug-fixes to the Keithley 2000, SR830, and Agilent E4408B
- Re-introduced the Newport ESP300 motion controller
- Major update to the Keithely 2400, 2000 and Yokogawa 7651 to achieve a common interface
- · New command-string processing hooks for Instrument property functions
- Updated LakeShore 331 temperature controller with new features
- Updates to the Agilent 8257D signal generator for better feature exposure

## 14.22 Version 0.4.1 - released 7/31/16

• Critical fix in setup.py for importing instruments (also added to documentation)

## 14.23 Version 0.4 – released 7/29/16

- · Replaced Instrument add\_measurement and add\_control with measurement and control functions
- Added validators to allow Instrument.control to match restricted ranges
- Added mapping to Instrument.control to allow more flexible inputs
- Conda is now used to set up the Python environment
- · macOS testing in continuous integration
- Major updates to the documentation

## 14.24 Version 0.3 - released 4/8/16

- Added IPython (Jupyter) notebook support with significant features
- Updated set of example scripts and notebooks
- New PyMeasure logo released
- Removed support for Python <3.4
- Changed multiprocessing to use spawn for compatibility
- Significant work on the documentation
- · Added initial tests for non-instrument code
- Continuous integration setup for Linux and Windows

## 14.25 Version 0.2 - released 12/16/15

- Python 3 compatibility, removed support for Python 2
- Considerable renaming for better PEP8 compliance
- · Added MIT License
- Major restructuring of the package to break it into smaller modules
- · Major rewrite of display functionality, introducing new Qt objects for easy extensions
- Major rewrite of procedure execution, now using a Worker process which takes advantage of multi-core CPUs
- Addition of a number of examples
- New methods for listening to Procedures, introducing ZMQ for TCP connectivity
- Updates to Keithley2400 and VISAAdapter

## 14.26 Version 0.1.6 - released 4/19/15

- Renamed the package to PyMeasure from Automate to be more descriptive about its purpose
- Addition of VectorParameter to allow vectors to be input for Procedures
- Minor fixes for the Results and Danfysik8500

## 14.27 Version 0.1.5 – release 10/22/14

- New Manager class for handling Procedures in a queue fashion
- New Browser that works in tandem with the Manager to display the queue
- · Bug fixes for Results loading

### 14.28 Version 0.1.4 – released 8/2/14

- Integrated Results class into display and file writing
- · Bug fixes for Listener classes
- Bug fixes for SR830

## 14.29 Version 0.1.3 – released 7/20/14

- Replaced logging system with Python logging package
- Added data management (Results) and bug fixes for Procedures and Parameters
- Added pandas v0.14 to requirements for data management
- · Added data listeners, Qt4 and PyQtGraph helpers

# 14.30 Version 0.1.2 - released 7/18/14

- Bug fixes to LakeShore 425
- Added new Procedure and Parameter classes for generic experiments
- Added version number in package

## 14.31 Version 0.1.1 - released 7/16/14

- Bug fixes to PrologixAdapter, VISAAdapter, Agilent 8722ES, Agilent 8257D, Stanford SR830, Danfysik8500
- Added Tektronix TDS 2000 with basic functionality
- Fixed Danfysik communication to handle errors properly

## 14.32 Version 0.1.0 - released 7/15/14

· Initial release

**CHAPTER** 

# **FIFTEEN**

# **RELATED PROJECTS**

PyMeasure is part of a broader ecosystem of laboratory automation tools:

- · LECO Protocol: Open standard for laboratory equipment communication using modern protocols
- pyleco: Python library implementing LECO for distributed instrument control systems
- Python Lab Automation Landscape: Curated collection of Python tools for laboratory automation and instrumentation

These projects provide complementary capabilities to PyMeasure for building robust laboratory automation systems.

PyMeasure Documentation, Release 0.15.1.dev366+gcb643a9c2.d20250827				

## **PYTHON MODULE INDEX**

```
р
                                               pymeasure.instruments, 97
                                               pymeasure.instruments.activetechnologies, 111
pymeasure.display.browser, 75
                                               pymeasure.instruments.aculight, 116
pymeasure.display.console, 75
                                               pymeasure.instruments.advantest, 117
pymeasure.display.curves, 76
                                               pymeasure.instruments.advantest.advantestR3767CG,
pymeasure.display.inputs, 77
pymeasure.display.listeners, 79
                                               pymeasure.instruments.advantest.advantestR624X,
pymeasure.display.log, 79
                                                       139
pymeasure.display.manager, 79
                                               pymeasure.instruments.agilent, 149
pymeasure.display.plotter, 81
                                               pymeasure.instruments.agilent.agilent4156,
pymeasure.display.thread, 81
pymeasure.display.widgets.browser_widget, 82
                                               pymeasure.instruments.agilent.agilentB1500,
pymeasure.display.widgets.directory_widget,
                                                       194
                                               pymeasure.instruments.aimtti, 215
pymeasure.display.widgets.dock_widget, 88
                                               pymeasure.instruments.aja, 218
pymeasure.display.widgets.estimator_widget,
                                               pymeasure.instruments.ametek, 220
                                               pymeasure.instruments.ami, 222
pymeasure.display.widgets.fileinput_widget,
                                               pymeasure.instruments.anaheimautomation, 224
                                               pymeasure.instruments.anapico, 226
pymeasure.display.widgets.filename_widget,83
                                               pymeasure.instruments.andeenhagerling, 227
pymeasure.display.widgets.image_frame, 83
                                               pymeasure.instruments.anritsu, 230
pymeasure.display.widgets.image_widget, 83
                                               pymeasure.instruments.attocube, 245
pymeasure.display.widgets.inputs_widget, 84
                                               pymeasure.instruments.bkprecision, 248
pymeasure.display.widgets.log_widget, 84
                                               pymeasure.instruments.comedi, 110
pymeasure.display.widgets.plot_frame, 84
                                               pymeasure.instruments.danfysik, 249
pymeasure.display.widgets.plot_widget, 84
                                               pymeasure.instruments.deltaelektronika, 252
pymeasure.display.widgets.results_dialog, 85
                                               pymeasure.instruments.edwards, 253
pymeasure.display.widgets.sequencer_widget,
                                               pymeasure.instruments.eurotest, 253
                                               pymeasure.instruments.fluke, 256
pymeasure.display.widgets.tab_widget, 87
                                               pymeasure.instruments.fwbell, 257
pymeasure.display.widgets.table_widget, 88
pymeasure.display.windows.managed_dock_window,pymeasure.instruments.hcp, 260
                                               pymeasure.instruments.heidenhain, 259
{\tt pymeasure.display.windows.managed\_image\_window}, {\tt pymeasure.instruments.hp}, 262
                                               pymeasure.instruments.inficon, 319
                                               pymeasure.instruments.ipgphotonics, 321
pymeasure.display.windows.managed_window, 91
                                               pymeasure.instruments.keithley, 323
pymeasure.display.windows.plotter_window, 94
                                               pymeasure.instruments.kepco, 407
pymeasure.experiment.experiment, 63
                                               pymeasure.instruments.keysight, 408
pymeasure.experiment.listeners, 64
                                               pymeasure.instruments.kuhneelectronic, 441
pymeasure.experiment.parameters, 67
                                               pymeasure.instruments.lakeshore, 444
pymeasure.experiment.procedure, 65
                                               pymeasure.instruments.lecroy, 455
pymeasure.experiment.results,72
                                               pymeasure.instruments.mksinst, 471
pymeasure.experiment.workers, 71
```

```
pymeasure.instruments.newport, 476
pymeasure.instruments.ni, 478
pymeasure.instruments.novanta, 492
pymeasure.instruments.oxfordinstruments, 493
pymeasure.instruments.parker, 505
pymeasure.instruments.pendulum, 507
pymeasure.instruments.philips, 506
pymeasure.instruments.proterial, 509
pymeasure.instruments.ptw, 511
pymeasure.instruments.racal, 519
pymeasure.instruments.razorbill, 521
pymeasure.instruments.redpitaya, 522
pymeasure.instruments.rigol, 524
pymeasure.instruments.rohdeschwarz, 526
pymeasure.instruments.santec, 544
pymeasure.instruments.siglenttechnologies,
pymeasure.instruments.signalrecovery, 551
pymeasure.instruments.spellmanhv, 564
pymeasure.instruments.srs, 568
pymeasure.instruments.tcpowerconversion, 585
pymeasure.instruments.tdk, 589
pymeasure.instruments.tektronix, 598
pymeasure.instruments.teledyne, 599
pymeasure.instruments.temptronic, 613
pymeasure.instruments.texio, 622
pymeasure.instruments.thermotron, 625
pymeasure.instruments.thorlabs, 626
pymeasure.instruments.thyracont, 627
pymeasure.instruments.toptica, 633
pymeasure.instruments.validators, 108
pymeasure.instruments.velleman, 635
pymeasure.instruments.yokogawa, 637
pymeasure.test, 57
```

714 Python Module Index

## **INDEX**

Symbols	ac_current (pymeasure.instruments.agilent.Agilent4284A
call() (pymeasure.instruments.agilent.agilentB1500 method), 193	ac_current(pymeasure.instruments.agilent.AgilentE4980
str() (pymeasure.instruments.agilent.agilentB1500. method), 194	O.CustomIntBronerty), 154 ac_mode() (pymeasure.instruments.lakeshore.LakeShore425
_format_binary_values() (pymea- sure.adapters.PrologixAdapter method), 54	7. / 1
_format_binary_values() (pymea- sure.adapters.SerialAdapter method), 51	ac_voltage (pymeasure.instruments.agilent.AgilentE4980 property), 154
Α	acceleration (pymea- sure.instruments.parker.ParkerGV6 property), 505
A (pymeasure.instruments.hp.hp856Xx.Trace attribute), 301	acq_buffer_filled (pymea-
<pre>abort() (pymeasure.display.console.ManagedConsole</pre>	property), 523  acq_format (pymeasure.instruments.redpitaya.redpitaya_scpi.RedPitaya
method), 79 abort() (pymeasure.instruments.agilent.agilentB1500.Ag	property), 523
abort() (pymeasure.instruments.agilent.AgilentB2981	property) 523
method), 201 abort() (pymeasure.instruments.agilent.AgilentE5062A method), 156	property), 323
abort() (pymeasure.instruments.anritsu.AnritsuMS20904	sure.instruments.reapitaya.reapitaya_scpi.ReaPitayaScpi
abort() (pymeasure.instruments.yokogawa.aq6370series. method), 640	sure.instruments.reapitava.reapitava_scpi.ReaPitavaScpi
abort_acquisition() (pymea- sure.instruments.agilent.AgilentB2981	property), 523  acq_trigger_source (pymea-
method), 201 abort_transient() (pymea- sure.instruments.agilent.AgilentB2985	property), 323
method), 207	acq_trigger_status (pymea- sure.instruments.redpitaya.redpitaya_scpi.RedPitayaScpi
sure.instruments.anaheimautomation.DPSeriesN property), 224	MotorController acq_units(pymeasure.instruments.redpitaya.redpitaya_scpi.RedPitayaS
absolute_to_steps() (pymea- sure.instruments.anaheimautomation.DPSeriesM method), 224	acquire_digital_input_output() (pymea- MotorController sure.instruments.ni.virtualbench.VirtualBench
AC (pymeasure.instruments.hp.hp856Xx.CouplingMode attribute) 301	method), 489 acquire_digital_multimeter() (pymea-

	sure.instruments.ni.virtualbench.Virtualbenc	alBench		sure.instruments.lecroy.LeCroyT3DSO property), 458	01204
acquire	_function_generator()	(pymea-	activat	ce() (pymeasure.instruments.agilent.agi	ilentE5062A.VNAChannel
	sure.instruments.ni.virtualbench.Virtua	ивепсп		method), 158	:loutE5062A VNATugo
2001110	method), 490	(12321111 0 0	activat	te() (pymeasure.instruments.agilent.agi	iieniE3002A.vivA1race
acquire	_mixed_signal_oscilloscope() sure.instruments.ni.virtualbench.Virtua		20111121	method), 160	nitauMC161xP Magaunama
		ивенсн	activat	te() (pymeasure.instruments.anritsu.an	tusum5404x <b>D</b> .measutemet
	method), 490	(		method), 242	witau MC16 Au D. Tugo o
acquire	_power_supply() sure.instruments.ni.virtualbench.Virtua	(pymea- alBench		e() (pymeasure.instruments.anritsu.an. method), 245	rusuws404xB.Trace
	method), 490		activat	te_auto_range()	(рутеа-
acquire	_reference()	(pymea-		sure.instruments.hp.HP437B method),	311
	sure.instruments.keithley.Keithley 2000		activat	ce_channel	(рутеа-
	method), 323			sure.instruments.rohdeschwarz.fsseries	s.FSW
acquire	_relative()	(pymea-		property), 541	
	sure.instruments.keithley.KeithleyDMN method), 383	16500	activat	e_source_peak_tracking() sure.instruments.hp.HP8560A method	(pymea- ) 296
acqui re		(рутеа-	active		(pymea-
acquire,				ebure.instruments.anritsu.AnritsuMS464	
	method), 399	11Cillicy21	02 Chann	property), 238	TAB
acquire	_voltage_reference()	(рутеа-	activo		(рутеа-
acquire				ebure.instruments.keithley.Keithley2182	4.0
	method), 399	Kenney21	02Cnunn	erty), 395	. ргор-
acquici	tion_average	(рутеа-	activo		(mymag
acquisi	_		active_	sure.instruments.rohdeschwarz.fsseries	(pymea-
	sure.instruments.lecroy.LeCroyT3DSO	1204			S.F.SW
	<pre>property), 458 tion_mode</pre>	(mymaa	activo	property), 541 connectors	(mymag
acquisi	sure.instruments.keysight.KeysightDSC		active_		(pymea- property),
		)X1102G		sure.instruments.hp.HP3478A p 268	торену),
	property), 408	(123,122,00	20+1110		C nuon
acquisi	tion_sample_size()		active_	gun (pymeasure.instruments.aja.DCX	.s prop-
	sure.instruments.lecroy.LeCroyT3DSO	1204	active_	erty), 218	(m)maga
	method), 458	(12321111 0 0	active_		(pymea-
acquisi	tion_sample_size_c1	(pymea-		sure.instruments.anritsu.AnritsuMS209	90A
	sure.instruments.lecroy.LeCroyT3DSO	1204	active_	property), 233	(m)maga
	<pre>property), 458 tion_sample_size_c2</pre>	(рутеа-	active_		(pymea-
acquisi	_			sure.instruments.agilent.Agilent4294A	prop-
	sure.instruments.lecroy.LeCroyT3DSO	1204		erty), 171	(77,000,000
	property), 458	(	active_	sure.instruments.anritsu.anritsuMS464	(pymea-
acquisi	tion_sample_size_c3	(pymea-			txB.MeasurementChannet
	sure.instruments.lecroy.LeCroyT3DSO	1204		property), 242	(77,000,000
	property), 458	(	active_		(pymea-
acquisi	tion_sample_size_c4	(pymea-		sure.instruments.yokogawa.aq6370ser	ies.AQ05/Oseries
	sure.instruments.lecroy.LeCroyT3DSO	1204		property), 640	
	property), 458	(	activit	cy (pymeasure.instruments.oxfordinstrum	nents.1P3120_10
acquisi	tion_sampling_rate	(pymea-	1	property), 499	14 DOD4CI   1
	sure.instruments.lecroy.LeCroyT3DSO	1204	actua1_	_flow(pymeasure.instruments.proterial.	.roa4.ROD4Cnannei
	property), 458	,		property), 511	
acquisi	tion_status	(pymea-	_	c (class in pymeasure.adapters), 47	70
	sure.instruments.lecroy.LeCroyT3DSO property), 458	1204	adc1 (p)	ymeasure.instruments.ametek.Ametek72 erty), 220	/U prop-
acquisi	tion_type	(pymea-	adc1 (	pymeasure.instruments.signalrecovery.l	DSP7225
	sure.instruments.key sight. Key sight DSC	OX1102G		property), 551	
	property), 408		adc1 (	pyme a sure. instruments. signal recovery. It is a surface of the surface of th	DSP7265
acquisi	tion_type	(pymea-		property), 557	

adc1 (pymeasure.instruments.srs.SR830 property), 575 adc1 (pymeasure.instruments.srs.SR860 property), 579	add_ramp_step() (pymea- sure.instruments.danfysik.Danfysik8500
adc2 (pymeasure.instruments.ametek.Ametek7270 property), 220	method), 249 add_smu() (pymeasure.instruments.keithley.Keithley4200
adc2 (pymeasure.instruments.signalrecovery.DSP7225	method), 405
property), 551	add_temperature_sensor() (pymea-
adc2 (pymeasure.instruments.signalrecovery.DSP7265 property), 558	sure.instruments.oxfordinstruments.MercuryiTC method), 502
adc2 (pymeasure.instruments.srs.SR830 property), 575	$address \ (py measure. instruments. an a heim automation. DPS eries Motor Continuous and Conti$
adc2 (pymeasure.instruments.srs.SR860 property), 579	property), 224
adc3 (pymeasure.instruments.ametek.Ametek7270 property), 220	address (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38 property), 589
adc3 (pymeasure.instruments.signalrecovery.DSP7265 property), 558	address (pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65 property), 594
adc3 (pymeasure.instruments.srs.SR830 property), 575	adjust_all() (pymea-
adc3 (pymeasure.instruments.srs.SR860 property), 579	sure.instruments.hp.hp856Xx.HP856Xx
adc3_time(pymeasure.instruments.signalrecovery.DSP72	
property), 558	<pre>adjust_if(pymeasure.instruments.hp.hp856Xx.HP856Xx</pre>
adc4 (pymeasure.instruments.ametek.Ametek7270 prop-	attribute), 279
erty), 220	AdvantestR3767CG (class in pymea-
adc4 (pymeasure.instruments.srs.SR830 property), 575	sure.instruments.advantest.advantestR3767CG),
adc4 (pymeasure.instruments.srs.SR860 property), 579	117
adc_auto_zero (pymea-	AdvantestR6245 (class in pymea-
sure.instruments.agilent.agilentB1500.AgilentB1. property), 187	500 sure.instruments.advantest.advantestR624X), 118
	AdvantestR6246 (class in pymea-
sure.instruments.agilent.agilentB1500.AgilentB1. method), 187	· · · · · · · · · · · · · · · · · · ·
adc_setup() (pymeasure.instruments.agilent.agilentB150	
method), 187	sure.instruments.advantest.advantestR624X),
adc_type(pymeasure.instruments.agilent.agilentB1500.SI	
property), 190	AFG3152C (class in pymeasure.instruments.tektronix),
ADCMode (class in pymea-	598
sure.instruments.agilent.agilentB1500), 195	AFG3152CChannel (class in pymea-
ADCType (class in pymea-	sure.instruments.tektronix.afg3152c), 598
sure.instruments.agilent.agilentB1500), 194	Agilent33220A (class in pymea-
add() (pymeasure.display.browser.Browser method), 75	sure.instruments.agilent), 172
add_child() (pymeasure.instruments.common_base.Com method), 98	n <b>AgiBæs</b> t 33500 (class in pymeasure.instruments.agilent), 174
add_child() (pymeasure.instruments.keithley.Keithley220	OAgilent33500Channel (class in pymea-
method), 332	sure.instruments.agilent.agilent33500), 178
add_child() (pymeasure.instruments.keysight.Keysight81 method), 431	MGALent 33521A (class in pymea- sure.instruments.agilent), 178
add_child() (pymeasure.instruments.keysight.KeysightE3	
method), 414	sure.instruments.agilent), 161
add_child() (pymeasure.instruments.keysight.KeysightE3 method), 422	sure.instruments.agilent), 161
${\tt add\_child()}$ (pymeasure.instruments.lecroy.LeCroyT3DS	•
method), 458	sure.instruments.agilent.agilent4156), 165
add_heater() (pymea-	Agilent4284A (class in pymeasure.instruments.agilent),
sure.instruments.oxfordinstruments.MercuryiTC	197
method), 502	Agilent4284ACorrection (class in pymea-
add_node() (pymeasure.display.widgets.sequencer_widge	
method), 86	Agilent4284ASpot (class in pymea-

sure.instruments.agilent.agilent4284A), 200	sure. instruments. lake shore. Lake Shore 421
${\tt Agilent 4294A} \ ({\it class in pymeasure.instruments.agilent}),$	property), 451
171	alarm_low_raw (pymea-
Agilent8257D (class in pymeasure.instruments.agilent),	sure.instruments.lakeshore.LakeShore421
149	property), 451
Agilent8722ES (class in pymea-	alarm_mode_enabled (pymea-
sure.instruments.agilent), 151	sure.instruments.lakeshore.LakeShore421
AgilentB1500 (class in pymea-	property), 451
sure.instruments.agilent.agilentB1500), 184	alarm_sort_enabled (pymea-
AgilentB2981 (class in pymeasure.instruments.agilent),	sure.instruments.lakeshore.LakeShore421
201	property), 451
AgilentB2983 (class in pymeasure.instruments.agilent),	all_outputs_enabled (pymea-
207	sure.instruments.aimtti.aimttiPL.PLBase
AgilentB2985 (class in pymeasure.instruments.agilent), 207	property), 215
	all_pressures (pymea- sure.instruments.mksinst.mks937b.MKS937B
AgilentB2987 (class in pymeasure.instruments.agilent), 213	property), 474
	all_values (pymeasure.instruments.inficon.sqm160.SQM160
AgilentE4408B (class in pymea- sure.instruments.agilent), 153	property), 320
AgilentE4980 (class in pymeasure.instruments.agilent),	am_depth (pymeasure.instruments.hp.HP8657B prop-
154	erty), 305
AgilentE5062A (class in pymea-	am_source (pymeasure.instruments.hp.HP8657B prop-
sure.instruments.agilent), 155	erty), 305
AgilentE5270B (class in pymea-	Ametek7270 (class in pymeasure.instruments.ametek),
sure.instruments.agilent), 213	220
AgilentMeasurementChannel (class in pymea-	AMI430 (class in pymeasure.instruments.ami), 222
sure.instruments.agilent.agilent4156), 168	amp_dbm (pymeasure.instruments.tektronix.afg3152c.AFG3152CChannel
AH2500A (class in pymea-	property), 599
sure.instruments.andeenhagerling), 227	amp_vpp (pymeasure.instruments.tektronix.afg3152c.AFG3152CChannel
AH2700A (class in pymea-	property), 599
sure.instruments.andeenhagerling), 228	amp_vrms (pymeasure.instruments.tektronix.afg3152c.AFG3152CChanne
air_temperature (pymea-	property), 599
sure.instruments.temptronic.ATSBase prop-	amplitude (pymeasure.instruments.agilent.Agilent33220A
erty), 613	property), 172
alarm_active (pymea-	amplitude (pymeasure.instruments.agilent.Agilent33500
sure.instruments.lakeshore.LakeShore421	property), 175
property), 451	amplitude (pymeasure.instruments.agilent.agilent33500.Agilent33500Cl
alarm_audible (pymea-	property), 178
sure. instruments. lake shore. Lake Shore 421	amplitude (pymeasure.instruments.hp.HP33120A prop-
property), 451	erty), 262
$\verb alarm_high  (pymeasure.instruments.lakeshore.LakeShore) $	e4 <b>2m</b> plitude (pymeasure.instruments.hp.HP8116A prop-
property), 451	erty), 271
alarm_high_multiplier (pymea-	${\tt Amplitude} \ (\textit{pymeasure.instruments.hp.hp856} Xx. Demodulation Mode$
sure. instruments. lake shore. Lake Shore 421	attribute), 301
property), 451	amplitude(pymeasure.instruments.keysight.Keysight81160A
alarm_high_raw (pymea-	property), 432
sure.instruments.lakeshore.Lake Shore 421	$amplitude \ (pymeasure. instruments. teledyne. teledyneT3AFG. Signal Chandle \ (pymeasure. instruments. teledyne. teledyneT3AFG. Signal Chandle \ (pymeasure. instruments. teledyne. teledyneT3AFG. Signal Chandle \ (pymeasure. instruments. teledyne. teledyne. teledyneT3AFG. Signal Chandle \ (pymeasure. instruments. teledyne. teledyne.$
property), 451	property), 601
alarm_in_out (pymea-	amplitude_depth (pymea-
sure.instruments.lakeshore.LakeShore421	sure.instruments.agilent.Agilent8257D prop-
property), 451	erty), 149
alarm_low(pymeasure.instruments.lakeshore.LakeShore4	
property), 451	sure.instruments.agilent.Agilent8257D prop-
alarm_low_multiplier (pymea-	erty), 149

amplitude_unit	(рутеа-		sure.instruments	s.agilent.agile	nt4156.Ag	gilent4156
sure.instruments.agilent.Agilent33220	A		property), 167			
property), 173		ANC300C	Controller	(class	in	рутеа-
amplitude_unit	(pymea-		sure.instruments	s.attocube.anc	300), 246	
sure.instruments.agilent.Agilent33500 erty), 175	prop-	angle (	pymeasure.instru erty), 505	ments.parker.l	ParkerGV	6 prop-
amplitude_unit	(рутеа-	angle_e	error (pymeasure	e.instruments. <sub>l</sub>	oarker.Pa	rkerGV6
sure.instruments.agilent.agilent33500.				•		
property), 178		annotat	ion_enabled			(рутеа-
amplitude_unit	(рутеа-		sure.instruments	s.hp.hp856Xx.	HP856Xx	at-
sure.instruments.hp.hp856Xx.HP856X	x at-		tribute), 280			
tribute), 275		Anritsu	MG3692C	(class	in	рутеа-
amplitude_unit	(рутеа-		sure.instruments	s.anritsu), 230		
sure.instruments.keysight.Keysight811	60A	Anritsu	MS2090A	(class	in	рутеа-
property), 432			sure.instrument.	s.anritsu), 233		
amplitude_units	(pymea-	Anritsu	MS4642B	(class	in	рутеа-
sure.instruments.hp.HP33120A p	property),		sure.instrument.	s.anritsu), 237		
262		Anritsu		`	in	рутеа-
AmplitudeUnits (class in	рутеа-		sure.instrument.	s.anritsu), 237		
sure.instruments.hp.hp856Xx), 300		Anritsu	MS4645B	(class	in	рутеа-
analog_configuration	(pymea-		sure.instrument.	s.anritsu), 237		
sure.instruments.lakeshore.LakeShore.	211	Anritsu	MS4647B	(class	in	рутеа-
property), 445			sure.instruments	s.anritsu), 237		
$\verb"analog_in" (pymeasure. instruments. redpit ay a.redpit ay a.re$	edpitaya_so	phinRádBiú	M\$4\$64xiB	(class	in	рутеа-
attribute), 522			sure.instruments	s.anritsu), 237		
analog_in_slow	(pymea-			`	in	рутеа-
sure.instruments.redpitaya.redpitaya	scpi.RedPii					
attribute), 522		Anritsu	MS9740A	(class	in	рутеа-
analog_input	(рутеа-		sure.instruments			
sure.instruments.advantest.advantestR	624X.SMU	(G/pæntedir		truments.keith	ley.Keith	leyDMM6500
property), 127			property), 384			
analog_monitor	(рутеа-	apertur	e() (pymeasure.	instruments.ag	gilent.Agi	lentE4980
sure.instruments.spellmanhv.Spellmannum	XRV		method), 154			
property), 564		append(		sure.display.cı	ırves.Buff	erCurve
analog_monitor	(рутеа-		method), 76			
sure.instruments.spellmanhv.spellman	XRV.Unsca	<i>l<b>aф</b>фћи</i> са				(pymea-
property), 567				s.anritsu.anrit	suMS464.	xB.MeasurementChanne
analog_out (pymeasure.instruments.lakeshore.	LakeShore					
property), 445		applied	l (pymeasure.inst	ruments.keithl	ey.Keithle	ey2260B
analog_out_slow	(рутеа-		property), 335	_		*****
sure.instruments.redpitaya.redpitaya_,	scpi.RedPii	tanpor Liped		ruments.texio.	TexioPSW	/360L30
attribute), 522		_	property), 622			
analog_output_setting			urrent()			(рутеа-
sure.instruments.thyracont.smartline_	v2.Smartlir	ıeV2	sure.instruments	s.keithley.Keiti	hley2400	
property), 629	,	_	method), 344			,
analog_reset()	* *		urrent()			(рутеа-
sure.instruments.redpitaya.redpitaya	scpi.RedPii	tayaScpi		s.keithley.Keiti	hley2450	
method), 523	1.00510	a .	method), 352			,
analysis (pymeasure.instruments.anritsu.Anrit	suMS9/10	Capply_c		7 77		(pymea-
property), 230	(		sure.instruments	s.yokogawa.Yo	кодаwa7	031
analysis_result	(pymea-		<i>method</i> ), 637	· · · · · · · · · · · · · · · · · · ·		
sure.instruments.anritsu.AnritsuMS97	10C	app1y_d		instruments.ke	ysight.ke	ysight81160A.Keysight8
property), 231	(		method), 439			(
analyzer_mode	(pymea-	apply_n	101Se()			(pymea-

			ght.keysight81	160A.Keysig	gl <b>a&amp;bl@W&amp;haæ(@l</b> ymeasure.instruments.keysight.	Keysight81160A
1	method)	, 439		(	property), 432	422500
apply_p			1.1 : 1.01		arb_file (pymeasure.instruments.agilent.Agil	ent33300
			gnt.keysignt81	100A.Keysig	ght81160AChapeety), 175	22500 A 3 22500Cl
	method)		. 1 . 1	.1 .1.01	arb_file (pymeasure.instruments.agilent.agilen	1133300.Agilen133300Cnan
app1y_s			ments.keysigh	t.keysight81	160A.Keysighupel160ACRannel	. 1.011604
-	method)	, 440		,	arb_file(pymeasure.instruments.keysight.Keys	signt81100A
apply_s	_		1.1 .1.01	(pymea-	property), 432	4 .22500
	sure.inst method)		ght.keysight81	160A.Keysig	gh <b>a&amp;bLGflACkan(py</b> measure.instruments.agilent.Ag property), 175	ilent33300
apply_u	ser_wav	eform()		(рутеа-	<pre>arb_filter(pymeasure.instruments.agilent.ag</pre>	ilent33500.Agilent33500Cl
	sure.inst	ruments.keysig	ght.keysight81	160A.Keysig	ght81160AG <b>haopeal</b> y), 178	
	method)	, 440			<pre>arb_filter(pymeasure.instruments.keysight.K</pre>	eysight81160A
apply_v	oltage(	)		(рутеа-	property), 432	
	sure.inst	ruments.keithl	ey.Keithley240	90	arb_srate(pymeasure.instruments.agilent.Agil	ent33500
	method)	, 344			property), 175	
apply_v	oltage(	)		(рутеа-	arb_srate(pymeasure.instruments.agilent.agil	ent33500.Agilent33500Cha
	sure.inst	ruments.keithl	ey.Keithley24.	50	property), 178	
	method)	, 352			arb_srate(pymeasure.instruments.agilent.Agil	ent33521A
apply_v	oltage(	)		(pymea-	property), 178	
	sure.inst	ruments.keithl	ey.Keithley65	17B	arb_srate(pymeasure.instruments.keysight.Ke	ysight81160A
	method)	, 378			property), 432	
apply_v	oltage(	)		(рутеа-	Argos (class in pymeasure.instruments.aculigh	it.argos),
	sure.inst	ruments.yokog	awa.Yokogaw	a7651	116	
	method)	, 637			arm() (pymeasure.instruments.agilent.Agile	entB2981
APSIN12	G (class i	n pymeasure.ir	istruments.and	apico), 227	method), 201	
AQ6370C		(class	in	рутеа-	arm() (pymeasure.instruments.siglenttechnologi	es.SDS1072CML
	sure.inst	ruments.yokog	awa.aq6370so	eries),	method), 548	
	643				<pre>arm_acquisition()</pre>	(рутеа-
AQ6370D		(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	
	sure.inst	ruments.yokog	awa.aq6370so	eries),	method), 201	
	643				arm_acquisition_bypass_once_enabled	(рутеа-
AQ6370E		(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	prop-
	sure.inst	ruments.yokog	awa.aq6370se	eries),	erty), 201	
	642				arm_acquisition_count	(рутеа-
AQ6370S	eries	(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	prop-
	sure.inst	ruments.yokog	awa.aq6370so	eries),	erty), 201	
	639				arm_acquisition_delay	(pymea-
AQ6373		(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	prop-
	sure.inst	ruments.yokog	awa.aq6370se	eries),	erty), 201	
	644				arm_acquisition_output_enabled	(рутеа-
AQ6373B		(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	prop-
	sure.inst	ruments.yokog	awa.aq6370se	eries),	erty), 201	
	645				arm_acquisition_output_signal	(pymea-
AQ6375		(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	prop-
	sure.inst	ruments.yokog	awa.aq6370se	eries),	erty), 202	
	646				arm_acquisition_source	(рутеа-
AQ6375B		(class	in	рутеа-	sure.instruments.agilent.AgilentB2981	prop-
	sure.inst	ruments.yokog	awa.aq6370se	eries),	erty), 202	
	646	, ,	_		arm_acquisition_source_lan_id	(рутеа-
arb_adv	ance (py	measure.instru	ments.agilent.	Agilent3350		• •
	property		-		erty), 202	
arb_adv	ance (pyr	measure.instru	ments.agilent.	agilent3350	00a <b>rmilæaddi160Chon</b> nalimer	(рутеа-
	property	178			sure instruments agilent AgilentR2981	pron-

erty), 202		ask() (pymeasure.instruments.aja.DCXS method	
		ask() (pymeasure.instruments.ametek.Am	etek7270
	prop-	method), 220	omm on Daga
erty), 202 arm_all_count (p:	утеа-	ask() (pymeasure.instruments.common_base.Comethod), 99	эттопъиѕе
	•	ask() (pymeasure.instruments.eurotest.Eurotest.	HPP120256
erty), 202	ргор	method), 255	111 1 120230
	ymea-	ask() (pymeasure.instruments.hp.HP8116A	method),
	prop-	272	,,
erty), 203		ask() (pymeasure.instruments.keysight.Keysigh	ut81160A
arm_all_output_enabled (p:	ymea-	<i>method</i> ), 432	
sure.instruments.agilent.AgilentB2981 erty), 203	prop-	ask() (pymeasure.instruments.keysight.Keysight method), 415	tE36312A
arm_all_output_signal (p	ymea-	$\verb"ask()" (pymeasure.instruments.keysight.Keysight]$	tE3631A
sure.instruments.agilent.AgilentB2981	prop-	<i>method</i> ), 423	
erty), 203		ask() (pymeasure.instruments.lecroy.LeCroyT31	DSO1204
	утеа-	method), 459	
sure.instruments.agilent.AgilentB2981 erty), 203	prop-	ask() (pymeasure.instruments.oxfordinstruments method), 493	s.base.OxfordInstrumentsB
4.	ymea-	$\verb"ask()" (pymeasure.instruments.thy racont.smartliff in the property of the $	ine_v2.SmartlineV2
	prop-	method), 629	
erty), 203		ask_manually()	(pymea-
	утеа-	sure.instruments.thyracont.smartline_v	2.SmartlineV2
	prop-	<pre>method), 630 at_temperature()</pre>	(mymag
erty), 204 arm_transient() (p:	утеа-	sure.instruments.temptronic.ATSBase	(pymea- method)
sure.instruments.agilent.AgilentB2985	утеа-	613	memoa),
method), 207		ATS525 (class in pymeasure.instruments.temptro	nic), 621
		ATS545 (class in pymeasure.instruments.temptro	
		ATSBase (class in pymeasure.instruments.tem	
erty), 208		613	
**	ymea-	$\verb attenuation   (pymeasure.instruments.agilent.agile$	gilentE5062A.VNAChannel
	prop-	property), 158	
erty), 208		attenuation (pymeasure.instruments.hp.hp856	<i>Xx.HP</i> 856 <i>Xx</i>
,	ymea-	attribute), 274	wan-fasarias ECCarias
erty), 208		$attenuation ({\it pymeasure.instruments.rohdesch} \\ {\it property}), 540$	
- ·		<pre>attenuation_x()</pre>	(pymea-
sure.instruments.agilent.AgilentB2985 erty), 208	prop-	sure.instruments.hp.HP11713A 310	method),
arm_transient_output_signal (p	ymea-	<pre>attenuation_y()</pre>	(pymea-
sure.instruments.agilent.AgilentB2985 erty), 208	prop-	sure.instruments.hp.HP11713A 310	method),
**	ymea-	Attenuator_110dB (in module	рутеа-
	prop-	sure.instruments.hp.hp11713a), 310	
erty), 208		Attenuator_11dB (in module	рутеа-
	утеа-	sure.instruments.hp.hp11713a), 310	
0 0	prop-	Attenuator_70dB_3_Section (in module	рутеа-
erty), 209 arm_transient_timer (p	утеа-	<pre>sure.instruments.hp.hp11713a), 310 Attenuator_70dB_4_Section (in module</pre>	рутеа-
	ymeu- prop-	sure.instruments.hp.hp11713a), 310	рушей-
erty), 209		authenticate_ethernet()	(рутеа-
ask() (pymeasure.instruments.agilent.agilentB1500		sure.instruments.yokogawa.aq6370seri	
method), 190		method), 640	~

$\verb"auto" (pymeasure. adapters. Prologix Adapter properties) and the properties of t$	erty), 54		<i>method</i> ), 323	
AUTO (pymeasure.instruments.agilent.agilentB150	0.ADCMo	<i>al</i> ato_ra:	nge()	(рутеа-
attribute), 195			sure.instruments.keithley.KeithleyDMM	M6500
AUTO (pymeasure.instruments.agilent.agilentB150	0.AutoMai	nual	method), 384	
attribute), 195		auto_ra	nge()	(pymea-
AUTO (pymeasure.instruments.agilent.agilentB150	0.Complia	ıncePolar	i <b>su</b> re.instruments.lakeshore.LakeShore	425
attribute), 197	•		method), 454	
AUTO (pymeasure.instruments.hp.hp856Xx.Amplita	udeUnits	auto_ra	nge_enabled	(рутеа-
attribute), 300			sure.instruments.agilent.Agilent4284A	prop-
auto_all() (pymeasure.instruments.rohdeschwa	rz.fsseries.	.FSW	erty), 198	
method), 541			nge_enabled	(pymea-
auto_calibration (	(рутеа-		sure.instruments.hp.HP3478A p	property),
sure.instruments.agilent.agilentB1500.A	AgilentB15	00	268	•
property), 186	-		nge_source()	(рутеа-
<pre>auto_freq() (pymeasure.instruments.rohdeschw</pre>			sure.instruments.keithley.Keithley2400	
method), 541	v		method), 344	
auto_gain (pymeasure.instruments.ametek.Amet	tek7270	auto_ra		(рутеа-
property), 220			sure.instruments.keithley.Keithley2450	= :
auto_gain(pymeasure.instruments.signalrecover	ry.DSP722	25	method), 352	
property), 551			nge_source()	(рутеа-
auto_gain(pymeasure.instruments.signalrecover			sure.instruments.keithley.Keithley6517	B
property), 558			method), 378	
	(рутеа-	auto_ra	nge_status()	(рутеа-
	operty),		sure.instruments.keithley.KeithleyDMM	* *
263	- I		method), 384	
	(pymea-	auto re	start_enabled	(рутеа-
sure.instruments.rohdeschwarz.fsseries.		_	sure.instruments.tdk.tdk_gen40_38.TD	4.5
method), 541			property), 590	
	(рутеа-		start_enabled	(рутеа-
sure.instruments.keithley.Keithley2182	A N	_	sure.instruments.tdk.tdk_gen80_65.TD	
method), 395			property), 594	
<pre>auto_offset() (pymeasure.instruments.srs</pre>	s.SR830		nsitivity()	(рутеа-
method), 575			sure.instruments.signalrecovery.DSP7.	* *
	(рутеа-		method), 551	
_	* *		nsitivity()	(рутеа-
erty), 344	r ·r		sure.instruments.signalrecovery.DSP7.	4.5
	(рутеа-		method), 558	
sure.instruments.signalrecovery.DSP722		auto_se		(pymea-
method), 551			sure.instruments.ni.virtualbench.Virtua	
	(рутеа-		method), 485	
sure.instruments.signalrecovery.DSP720		auto ze	ro (pymeasure.instruments.keithley.Kei	thlev2400
method), 558			property), 344	,=
auto_pid(pymeasure.instruments.oxfordinstrume	ents.ITC50	Buto ze:		(рутеа-
property), 494			sure.instruments.advantest.advantestR	
- · ·	(рутеа-		property), 138	
sure.instruments.oxfordinstruments.ITC.			ro_enabled	(pymea-
property), 495				property),
auto_range(pymeasure.instruments.lakeshore.Lak	akeShore4	21	268	T
property), 451			ro_enabled	(рутеа-
	(рутеа-		sure.instruments.keithley.Keithley2182	
	nethod),		erty), 395	I - I
257		AutoMan	• •	рутеа-
	(рутеа-		sure.instruments.agilent.agilentB1500	* *
sure.instruments.keithley.Keithley2000		automat	ic_range_enabled	(pymea-

sure.instruments.hp.HP437B proper	tu) 311	2117 O11+	_1 (pymeasure.instruments.srs.SR830 p	ronarty)
automatic_sample_number	(pymea-	aux_out	_1 (pymeasure.instruments.srs.sRo30 p 575	торену),
	* *	OSmaia ou+	_1 (pymeasure.instruments.srs.SR860 p	ronarty)
property), 640	eries.AQ057	Daemenut.	_1 (pymeasure.instruments.srs.sR600 p 579	торену),
* * · * ·	1014 nuon	2117 211+		man autu)
autorange (pymeasure.instruments.hp.HP344	+01A prop-	aux_out	_2 (pymeasure.instruments.srs.sRo30 p 575	торену),
erty), 264	(			
autorange_enabled	(pymea-	aux_out	_2 (pymeasure.instruments.srs.SR860 p	roperty),
sure.instruments.keithley.KeithleyDM	1M6300		579	. \
property), 384		aux_out	_3 (pymeasure.instruments.srs.SR830 p	roperty),
autorange_enabled	(pymea-	G 10	575	,
	1M6500.Scai	n <b>aeux_aou</b> tz	<b>QBQGymnasl</b> ıre.instruments.srs.SR860 p	roperty),
property), 393			579	
autoreset_enabled	(рутеа-	aux_out	_4 (pymeasure.instruments.srs.SR830 p	roperty),
sure.instruments.ptw.ptwUNIDOS	property),		575	
514			_4 (pymeasure.instruments.srs.SR860 p	roperty),
<pre>autoscale() (pymeasure.instruments.keysigh</pre>	t.KeysightD			
method), $408$			ry_condition_code	(рутеа-
autoscale() (pymeasure.instruments.lecroy.	LeCroyT3DS	SO1204	sure.instruments.temptronic.ATSB ase	prop-
method), 459			erty), 613	
$\verb"autoscale" () (pymeasure.instruments.teledyn$	e.teledyneM.	A <b>t</b> MaTielledby	hæ <u>MAM6elis</u> nel	(рутеа-
method), 612			sure.instruments.rohdeschwarz.fsseries	s.FSW
<pre>autoscale() (pymeasure.instruments.teledyn</pre>	e.TeledyneO	scilloscope	eproperty), 541	
method), 603		average.	_acceleration	(pymea-
autostart_enabled	(рутеа-		sure.instruments.parker.ParkerGV6 p	roperty),
sure.instruments.ptw.ptwUNIDOS	property),		505	•
514		average	_count	(pymea-
autostart_level	(рутеа-	J	sure.instruments.anritsu.anritsuMS464	
sure.instruments.ptw.ptwUNIDOS	property),		property), 242	
514	1 1 377	average		(pymea-
autovernier_enabled	(рутеа-		sure.instruments.anritsu.AnritsuMS97	4.7
sure.instruments.hp.HP8116A	property),		property), 231	
272	F: "F: ",",	average.		(рутеа-
autozero_enabled	(рутеа-	u 1 02 u 9 0.	sure.instruments.inficon.sqm160.SQM	* *
sure.instruments.hp.HP34401A	property),		property), 320	
264	property),		_skipped_samples	(рутеа-
autozero_enabled	(рутеа-	average.	_sure.instruments.redpitaya.redpitaya_s	4.5
sure.instruments.keithley.KeithleyDM			property), 523	еринеат науавері
property), 384	1110300			(pymea-
aux_in_1 (pymeasure.instruments.srs.SR830	nronarty)	average.	_sweep sure.instruments.anritsu.AnritsuMS971	= -
575	property),		property), 231	100
	nronarty)			(рутеа-
aux_in_1 (pymeasure.instruments.srs.SR860	property),	average	_	4.
579			sure.instruments.anritsu.AnritsuMS974	<i>tUA</i>
aux_in_2 (pymeasure.instruments.srs.SR830	property),		property), 233	(
575		average.	_sweep_count	(pymea-
aux_in_2 (pymeasure.instruments.srs.SR860	property),		sure.instruments.anritsu.anritsuMS464	xB.MeasurementCnanne
579			property), 242	/
aux_in_3 (pymeasure.instruments.srs.SR830	property),	average.	_thickness	(pymea-
575			sure.instruments.inficon.sqm160.SQM	160
aux_in_3 (pymeasure.instruments.srs.SR860	property),		property), 320	
579		average.		(pymea-
aux_in_4 (pymeasure.instruments.srs.SR830	property),		sure.instruments.anritsu.anritsuMS464	xB.MeasurementChanne
575			property), 242	050050
aux_in_4 (pymeasure.instruments.srs.SR860	property),		s (pymeasure.instruments.agilent.Agiler	nt8722ES
579			property), 151	

$averages \ (pymeasure. instruments. a gilent. a gilent E 5062A.$	
property), 158	sure. instruments. active technologies. AWG 401x. Channel AFG
averages (pymeasure.instruments.hp.HP8753E prop-	property), 114
erty), 307	BaseManager (class in pymeasure.display.manager), 79
averaging_enabled (pymea-	${\tt basespeed} \ (py measure. instruments. an a heim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. DPS eries Motor College \ (py measure. instruments. an aheim automation. and an alternative automation. and an alternative automation. Alternative$
sure.instruments.agilent.Agilent8722ES	property), 225
property), 152	basic_info(pymeasure.instruments.rohdeschwarz.sfm.SFM
averaging_enabled (pymea-	property), 527
	antoatch_size (pymeasure.instruments.pendulum.cnt91.CNT91
property), 158	property), 507
	battery_cycles (pymea- urementChasumed.instruments.agilent.agilentB298x.BatteryMixin
property), 243	property), 213
averaging_enabled (pymea-	battery_level (pymea-
sure.instruments.hp.HP8753E property),	sure.instruments.agilent.agilentB298x.BatteryMixin
307	property), 213
	battery_selftest_passed (pymea-
sure.instruments.hp.HP8753E method), 307	sure.instruments.agilent.agilentB298x.BatteryMixin
AWG401x_AFG (class in pymea- sure.instruments.activetechnologies), 111	<pre>property), 213 BatteryMixin (class in pymea-</pre>
AWG401x_AWG (class in pymea-	BatteryMixin (class in pymea- sure.instruments.agilent.agilentB298x), 213
sure.instruments.activetechnologies), 112	baud_rate(pymeasure.instruments.thyracont.smartline_v2.SmartlineV2
AWG401x_AWG.DummyEntriesElements (class in	- · · · · · · · · · · · · · · · · · · ·
pymeasure.instruments.activetechnologies),	baudrate (pymeasure.instruments.ptw.ptwDIAMENTOR
112	property), 512
AWG401x_AWG.WaveformsLazyDict (class in pymea-	baudrate (pymeasure.instruments.spellmanhv.SpellmanXRV
sure.instruments.activetechnologies), 112	property), 564
axes (pymeasure.instruments.newport.ESP300 property), 476	beep() (pymeasure.instruments.agilent.Agilent33220A method), 173
Axis (class in pymeasure.instruments.attocube.anc300),	beep() (pymeasure.instruments.agilent.Agilent33500
247	method), 175
Axis (class in pymeasure.instruments.newport.esp300), 477	beep() (pymeasure.instruments.agilent.Agilent34450A method), 161
AxisError (class in pymea-	beep() (pymeasure.instruments.hp.HP33120A method),
sure.instruments.newport.esp300), 477	262
В	beep() (pymeasure.instruments.hp.HP34401A method), 264
B (pymeasure.instruments.hp.hp856Xx.Trace attribute), 301	beep() (pymeasure.instruments.keithley.Keithley2000 method), 323
$bandwidth ({\it pymeasure.instruments.anritsu.anritsuMS464}$	1x <b>B9AP</b> asuremeynensureinstruments.keithley.Keithley2400 method), 344
property), 243 bandwidth_enhancer_enabled (pymea-	beep() (pymeasure.instruments.keithley.Keithley2450
sure.instruments.anritsu.AnritsuMS464xB	method), 352
property), 238	beep() (pymeasure.instruments.keithley.Keithley2700
BASE (pymeasure.instruments.agilent.agilentB1500.Sampli	ing PostOutsmethod), 364
attribute), 196	beep() (pymeasure.instruments.keithley.Keithley6221
BaseBrowserItem (class in pymea-	method), 371
sure.display.browser), 75	beep() (pymeasure.instruments.keithley.KeithleyDAQ6510
haseline offset (nymea-	method), 400
sure. instruments. active technologies. AWG 401x. C	Therea(A) (gymeasure.instruments.keithley.KeithleyDMM6500
property), 114	metnoa), 384
baseline_offset_max (pymea-	beep() (pymeasure.instruments.kepco.KepcoBOP3612
sure.instruments.activetechnologies.AWG401x.C	ChannelAFGmethoa), 40/
property), 114	beep() (pymeasure.instruments.keysight.Keysight81160A

4 1) 422			1 1 150			
method), 432	HMD 4		method), 459	( 1		
beep() (pymeasure.instruments.rohdeschwarz.hi	пр.НМР40			(class		рутеа-
method), 542	:4.1 2000		sure.instruments.i			CD570
beep_state (pymeasure.instruments.keithley.Ke	iiniey2000			ieasure.instri	iments.srs	.SK3/0
<pre>property), 323 beeper_enabled</pre>	(mymag	blank_ti	method), 573		(	nymaa
<del>-</del>	* *		sure.instruments.i	hn hn856Vr 1	,	рутеа-
264	roperty),		method), 288	пр.проэолх.1	11 03011	
	(nymea-		meinoa), 288 g (pymeasure.in	strumonts an	anico APS	SIN12G
sure.instruments.rohdeschwarz.sfm.SFI			property), 227	sir umenis.an	upico.111 L	111120
property), 527	,,			struments red	dnitava re	dpitaya_scpi.RedPitayaS
	(рутеа-		property), 524	sir umenis.rec	ap iici yei.i e	apitaya_sept.frear trayas
sure.instruments.agilent.Agilent33220A			Input ( <i>class in p</i> y	measure.dist	olav.inputs	s), 77
property), 173				(class		рутеа-
BIAS (pymeasure.instruments.agilent.agilentB15)	00.Samplii			*		F.J
attribute), 196	· · · · · · · · · · · · · · · · · · ·		der_version	.,,,		рутеа-
bias (pymeasure.instruments.srs.ldc500series.Ll	DC500Seri		sure.instruments.i	thyracont.sm		• •
property), 570			property), 630	•	_	
	(рутеа-		t (pymeasure.instr	uments.kepce	о.КерсоВ	OP3612
sure.instruments.agilent.Agilent4284A	prop-		property), 407	-	-	
erty), 198		both_cha	annels_enabled		(	рутеа-
bias_enabled	(рутеа-		sure.instruments.i	keithley.Keith	ley2306	prop-
sure.instruments.agilent.Agilent4284A	prop-		erty), 341			
erty), 198		${\tt Browser}$	(class in pymeasu	re.display.br	owser), 75	5
			Item ( <i>class in pyn</i>	neasure.displ	ay.browse	(r), 75
sure.instruments.kuhneelectronic.Kusg	245_250A					рутеа-
property), 442			sure.display.widg		_	
<pre>bias_enabled (pymeasure.instruments.srs.SR5) erty), 573</pre>	70 prop-		(pymeasure.instruattribute), 337	ıments.keithli	ey.Keithle	y2281S
bias_level (pymeasure.instruments.srs.SR576	) prop-	bt	(pymeasure.instru	ıments.keithl	ey.Keithle	y2281S
erty), 573			attribute), 337			
bias_voltage	(рутеа-	buffer_c	data ( <i>pymeasure.i</i>	instruments.k	eithley.Ke	rithley2000
sure.instruments.agilent.Agilent4284A	prop-		property), 323			
erty), 198			data( <i>pymeasure.i</i>	instruments.k	eithley.Ke	eithley2182
	(рутеа-		property), 395			
sure.instruments.anritsu.AnritsuMS464	AxB		data ( <i>pymeasure.i</i>	instruments.k	eithley.Ke	eithley2281S
property), 238			property), 337	_		
	(pymea-		data(pymeasure.i	instruments.k	eithley.Ke	tithley2400
sure.instruments.common_base.Commo	onBase		property), 344		1.1.1 77	0.450
method), 99	,		data(pymeasure.i	instruments.k	eithley.Ke	rithley2450
	(рутеа-		property), 352		.11 17	:41 2700
sure.instruments.keithley.Keithley2200 method), 332			data(pymeasure.i property), 364	nstruments.k	eithley.Ke	ithley2/00
binary_values()	(рутеа-	buffer_c	data (pymeasure.i	instruments.k	eithley.Ke	rithley6221
sure.instruments.keysight.Keysight8116	60A		property), 371			
method), 432		buffer_c	data ( <i>pymeasure.i</i>	instruments.k	eithley.Ke	rithley6517B
binary_values()	(рутеа-		property), 378			
sure.instruments.keysight.KeysightE36.	312A	buffer_c	data ( <i>pymeasure.i</i>	instruments.k	eithley.Ke	rithleyDAQ6510
method), 415			property), 400			
	(рутеа-		frequency_time			рутеа-
sure. instruments. key sight. Key sight E36.	31A		sure.instruments.j	pendulum.cni	91.CNT9	1
method), 423			method), 507			
		buffer_				рутеа-
sure.instruments.lecroy.LeCroyT3DSO	1204		sure.instruments.i	redpitaya.red	pitaya_sc	pi.RedPitayaScpi

property), 524	$burst\_mode (\textit{pymeasure.instruments.agilent.Agilent33220A}$
<pre>buffer_measure() (pymeasure.instruments.srs.SR830</pre>	property), 173
method), 575	burst_mode(pymeasure.instruments.agilent.Agilent33500
buffer_points (pymea-	property), 175
sure.instruments.keithley.Keithley2000 prop-	burst_mode(pymeasure.instruments.agilent.agilent33500.Agilent33500C
erty), 323	property), 178
buffer_points (pymea-	burst_mode(pymeasure.instruments.keysight.Keysight81160A
sure.instruments.keithley.Keithley2182 prop-	property), 432
erty), 395	burst_ncycles (pymea-
buffer_points (pymea-	sure.instruments.agilent.Agilent33220A
sure.instruments.keithley.Keithley2281S	property), 173
property), 337	burst_ncycles (pymea-
buffer_points (pymea-	sure.instruments.agilent.Agilent33500 prop-
sure.instruments.keithley.Keithley2400 prop-	erty), 176
erty), 344	burst_ncycles (pymea-
• •	
buffer_points (pymea-	sure.instruments.agilent.agilent33500.Agilent33500Channel
sure.instruments.keithley.Keithley2450 prop-	property), 178
erty), 353	burst_ncycles (pymea-
buffer_points (pymea-	sure.instruments.keysight.Keysight81160A
sure.instruments.keithley.Keithley2700 prop-	property), 432
erty), 365	burst_number (pymeasure.instruments.hp.HP8116A
buffer_points (pymea-	property), 272
sure.instruments.keithley.Keithley6221 prop-	burst_period (pymea-
erty), 371	sure.instruments.agilent.Agilent33500 prop-
buffer_points (pymea-	erty), 176
sure.instruments.keithley.Keithley6517B	burst_period (pymea-
property), 378	sure.instruments.agilent.agilent33500.Agilent33500Channel
buffer_points (pymea-	property), 178
sure.instruments.keithley.KeithleyDAQ6510	burst_period (pymea-
property), 400	sure.instruments.keysight.Keysight81160A
buffer_points (pymea-	property), 433
sure.instruments.keithley.KeithleyDMM6500	burst_phase (pymeasure.instruments.hp.HP33120A
property), 385	property), 262
buffer_size (pymeasure.instruments.keithley.KeithleyDi	
	property), 262
property), 385 buffer_to_float() (pymea-	
4.2	burst_source (pymeasure.instruments.hp.HP33120A
sure.instruments.signalrecovery.DSP7225	property), 262
method), 551	burst_state(pymeasure.instruments.agilent.Agilent33220A
buffer_to_float() (pymea-	property), 173
sure.instruments.signalrecovery.DSP7265	burst_state(pymeasure.instruments.agilent.Agilent33500
method), 558	property), 176
BufferCurve (class in pymeasure.display.curves), 76	burst_state(pymeasure.instruments.agilent.agilent33500.Agilent33500
$burst\_count (\textit{pymeasure.} instruments. active technologies.$	
property), 112	burst_state(pymeasure.instruments.keysight.Keysight81160A
burst_count (pymeasure.instruments.hp.HP33120A	property), 433
property), 262	busy (pymeasure.instruments.anaheimautomation.DPSeriesMotorControl
burst_count_max (pymea-	property), 225
sure.instruments.activetechnologies.AWG401x A	AVEGIimit (pymeasure.instruments.lecroy.LeCroyT3DSO1204
property), 112	property), 459
burst_count_min (pymea-	bwlimit (pymeasure.instruments.lecroy.lecroyT3DSO1204.LeCroyT3DSO
sure.instruments.activetechnologies.AWG401x_A	
property), 112	bwlimit (pymeasure.instruments.teledyne.teledyne_oscilloscope.Teledyne
burst_enabled (pymeasure.instruments.hp.HP33120A	property), 607
property), 262	bwlimit (pymeasure.instruments.teledyne.TeledyneOscilloscope
property), 202	bwrrmre (pymeusure.msnumems.wwayne.rewayneOscmoscope

С	property), 603		capacit	ance_range sure.instruments.agilent.Agilent34450.	(pymea- A
				property), 162	(
cable_l	ength	(рутеа-	capacit	ance_range	(pymea-
	sure.instruments.agilent.agilent4284A	Agilent428.	84ACorrec	tsyre.instruments.keithley.KeithleyDMM	V10300
_	property), 200			property), 385	(100,000,000
	sult (pymeasure.instruments.yokogaw	a.aq6370se	nepaque	mse <sub>ric</sub> elative	(pymea-
	property), 640	/		sure.instruments.keithley.KeithleyDMM	40300
calibra		.HP437B	canaci+	<pre>property), 385 ance_relative_status</pre>	(pwm ag
	method), 311				(pymea- M6500
calibra	te() (pymeasure.instruments.srs.ldc30	Oseries.LD	C500Seru	<sub>e</sub> s <b>pn</b> g.instruments.keithley.KeithleyDM1 property), 385	10300
	method), 570	,	canacit	y (pymeasure.instruments.attocube.and	300 Aris
calibra		(pymea-	Capacit	property), 247	JOO.Axis
	sure.instruments.rohdeschwarz.sfm.SF	M	canloss	volt (pymeasure.instruments.andeenho	gaerling AH2500A
	method), 528	,	Capioss	property), 227	igerung.A112500A
	tion_data	(pymea-	canloss	volt (pymeasure.instruments.andeenho	gaerling AH2700A
	•	property),	Capioss	property), 228	ige1111g.111270011
1:1	269	(	carrier	_enabled	(рутеа-
calibra	tion_enabled	(pymea-		asurg.instruments.rohdeschwarz.sfm.So	
		4x <b>B</b> .Measu	rementCn	property), 537	una_onamei
	property), 243	,	carrier	_frequency	(рутеа-
	tion_enabled	(pymea-	Callici	sure.instruments.rohdeschwarz.sfm.So	4.5
		property),		property), 537	una_Channei
1:1	269	,	carrier		(рутеа-
calibra	tion_factor	(pymea-			4.5
	property), 139	624X.SMU		sure.instruments.rohdeschwarz.sfm.So. property), 537	
calibra	tion_factor	(рутеа-	cathode	_enabled	(pymea-
	sure.instruments.hp.HP437B property	), 311		sure.instruments.thyracont.smartline_v	v1.SmartlineV1
calibra	tion_factor	(рутеа-		property), 628	
	sure.instruments.ptw.ptwDIAMENTOI	R prop-	celcius	(pymeasure.instruments.lakeshore.lake	shore_base.LakeShoreTem <sub>[</sub>
	erty), 512			property), 454	
calibra	tion_generation_factor	(рутеа-		(pymeasure.instruments.lakeshore.lake	shore_base.LakeShoreTem <sub>p</sub>
	sure. instruments. advantest. advantest R	624X.SMU	Channel	property), 454	
	property), 139		center_	at_peak()	(pymea-
calibra	tion_init()	(рутеа-		sure.instruments.anritsu.AnritsuMS97	10C
	sure. instruments. advantest. advantest R	624X.SMU	Channel	method), 231	
	method), 139		center_	frequency	(pymea-
calibra	tion_measured_value	(рутеа-		sure.instruments.advantest.advantestR	3/6/CG.AdvantestR3/6/C
	sure. instruments. advantest. advantest R	624X.SMU	Channel	property), 117	,
	property), 139		center_	frequency	(pymea-
calibra	tion_store_factor()	(рутеа-		sure.instruments.agilent.Agilent8257D	prop-
	sure. instruments. advantest. advantest R	624X.SMU	Channel	erty), 149	
	method), 139			frequency	(pymea-
capacit	${\sf ance} (pymeasure. instruments. agilent. A$	gilent3445	OA	sure.instruments.agilent.AgilentE4408	В
	property), 161			property), 153	
capacit	ance (pymeasure.instruments.keithley.	KeithleyDM	1 <i>M</i> 05500r_	frequency	(рутеа-
	property), 385			sure.instruments.hp.hp856Xx.HP856X	x at-
capacit	ance_auto_range	(рутеа-		tribute), 282	
	sure.instruments.agilent.Agilent34450		center_	frequency	(pymea-
	property), 161			•	property),
capacit	ance_digits	(рутеа-		307	
	sure.instruments.keithley.KeithleyDM	M6500	center_	trigger()	(рутеа-
	property), 385			sure.instruments.lecroy.LeCroyT3DSC	01204

method), 459 center_trigger() (pymea-	ch_1 (pymeasure.instruments.teledyne.TeledyneT3AFG attribute), 600
sure.instruments.teledyne.TeledyneOscilloscope method), 603	ch_1 (pymeasure.instruments.toptica.ibeamsmart.IBeamSmart attribute), 633
ch() (pymeasure.instruments.keithley.Keithley2306 method), 341	ch_2 (pymeasure.instruments.activetechnologies.AWG401x_AFG attribute), 111
ch() (pymeasure.instruments.lecroy.LeCroyT3DSO1204 method), 459	ch_2 (pymeasure.instruments.agilent.Agilent33500 at- tribute), 175
	pch_2 (pymeasure.instruments.agilent.Agilent33521A attribute), 178
ch_1 (pymeasure.instruments.activetechnologies.AWG401: attribute), 111	x_ <b>AF_Q</b> (pymeasure.instruments.agilent.AgilentE5062A attribute), 156
ch_1 (pymeasure.instruments.agilent.Agilent33500 attribute), 175	ch_2 (pymeasure.instruments.aimtti.aimttiPL.PL303QMDP attribute), 217
ch_1 (pymeasure.instruments.agilent.Agilent33521A attribute), 178	ch_2 (pymeasure.instruments.aimtti.aimttiPL.PL303QMTP attribute), 217
ch_1 (pymeasure.instruments.agilent.AgilentE5062A attribute), 156	ch_2 (pymeasure.instruments.keithley.Keithley2182 at- tribute), 395
ch_1 (pymeasure.instruments.aimtti.aimttiPL.PL068P attribute), 216	ch_2 (pymeasure.instruments.keithley.Keithley2200 at- tribute), 330
ch_1 (pymeasure.instruments.aimtti.aimttiPL.PL155P attribute), 216	ch_2 (pymeasure.instruments.keysight.Keysight81160A attribute), 430
ch_1 (pymeasure.instruments.aimtti.aimttiPL.PL303P attribute), 216	ch_2 (pymeasure.instruments.keysight.KeysightE36312A attribute), 413
ch_1 (pymeasure.instruments.aimtti.aimttiPL.PL303QMD attribute), 217	attribute), 421
attribute), 217	P ch_2 (pymeasure.instruments.lecroy.LeCroyT3DSO1204 attribute), 456
ch_1 (pymeasure.instruments.aimtti.aimttiPL.PL601P attribute), 216	ch_2 (pymeasure.instruments.mksinst.mks937b.MKS937B attribute), 472
ch_1 (pymeasure.instruments.keithley.Keithley2182 at- tribute), 394	ch_2 (pymeasure.instruments.proterial.ROD4 attribute), 509
ch_1 (pymeasure.instruments.keithley.Keithley2200 at- tribute), 330	ch_2 (pymeasure.instruments.teledyne.TeledyneMAUI attribute), 610
ch_1 (pymeasure.instruments.keysight.Keysight81160A attribute), 430	ch_2 (pymeasure.instruments.teledyne.TeledyneOscilloscope attribute), 603
ch_1 (pymeasure.instruments.keysight.KeysightE36312A attribute), 413	ch_2 (pymeasure.instruments.teledyne.TeledyneT3AFG attribute), 600
ch_1 (pymeasure.instruments.keysight.KeysightE3631A attribute), 421	ch_2 (pymeasure.instruments.toptica.ibeamsmart.IBeamSmart attribute), 633
ch_1 (pymeasure.instruments.lecroy.LeCroyT3DSO1204 attribute), 456	ch_3 (pymeasure.instruments.agilent.AgilentE5062A attribute), 156
ch_1 (pymeasure.instruments.mksinst.mks937b.MKS937B attribute), 472	ch_3 (pymeasure.instruments.aimtti.aimttiPL.PL303QMTP attribute), 217
ch_1 (pymeasure.instruments.proterial.ROD4 attribute), 509	ch_3 (pymeasure.instruments.keithley.Keithley2200 attribute), 330
attribute), 548	8Xh_3 (pymeasure.instruments.keysight.KeysightE36312A attribute), 413
attribute), 548	5Xh_3 (pymeasure.instruments.keysight.KeysightE3631A attribute), 421
ch_1 (pymeasure.instruments.teledyne.TeledyneMAUI attribute), 610	ch_3 (pymeasure.instruments.lecroy.LeCroyT3DSO1204 attribute), 456
ch_1 (pymeasure.instruments.teledyne.TeledyneOscillosco attribute), 602	p <b>c</b> h_3 (pymeasure.instruments.mksinst.mks937b.MKS937B attribute), 472

ch_3 (pymeasure.instruments.proterial.ROD4 attribute), 509	channel	property), 634 _1 (pymeasure.instruments.rigol.DC	5800 at-
ch_3 (pymeasure.instruments.teledyne.TeledyneMAUI attribute), 610		tribute), 5241 (pymeasure.instruments.siglenttech	
ch_3 (pymeasure.instruments.teledyne.TeledyneOscillosco attribute), 603	pe	attribute), 548	
ch_3 (pymeasure.instruments.toptica.ibeamsmart.IBeamSi		2 (pymeasure.instruments.rigol.DC tribute), 524	1000 ui-
attribute), 633	channel	_2 (pymeasure.instruments.siglenttech	nologies.SDS1072CML
ch_4 (pymeasure.instruments.agilent.AgilentE5062A attribute), 156	channel	<pre>attribute), 548down_relative()</pre>	(рутеа-
ch_4 (pymeasure.instruments.lecroy.LeCroyT3DSO1204 attribute), 456	Chamer	sure.instruments.rohdeschwarz.sfm.Si method), 528	
ch_4 (pymeasure.instruments.mksinst.mks937b.MKS937B attribute), 472	channel		(pymea- AgilentMeasurementChanne
ch_4 (pymeasure.instruments.proterial.ROD4 attribute), 509	channel	property), 168 _function	(рутеа-
ch_4 (pymeasure.instruments.teledyne.TeledyneMAUI attribute), 610		sure.instruments.keithley.Keithley218 erty), 395	2 prop-
ch_4 (pymeasure.instruments.teledyne.TeledyneOscillosco	p <b>c</b> hannel	_mode	(рутеа-
attribute), 603		sure.instruments.agilent.agilent4156	AgilentMeasurementChanne
ch_4 (pymeasure.instruments.toptica.ibeamsmart.IBeamSi		property), 168	(nym o a
attribute), 633 ch_5 (pymeasure.instruments.mksinst.mks937b.MKS937B	channel	_mode sure.instruments.agilent.agilent4156.	(pymea- VARD
attribute), 472		property), 170	WIND
ch_5 (pymeasure.instruments.toptica.ibeamsmart.IBeamSi	m <b>æh</b> annel	* * * ·	(рутеа-
attribute), 633		sure.instruments.agilent.agilent4156.	VARX
ch_6 (pymeasure.instruments.mksinst.mks937b.MKS937B		property), 171	(
attribute), 472 ch_A (pymeasure.instruments.advantest.advantestR624X.A		_settings()	(pymea- method),
attribute), 118	avamesiK	519	memoa),
ch_A (pymeasure.instruments.advantest.advantestR624X.A	d colorate sveRi	* - /	(рутеа-
attribute), 118		sure.instruments.rohdeschwarz.sfm.SI	FM
${\tt ch\_B} \ (py measure. instruments. advantest. advantest R 624 X. A$			
attribute), 118		_sweep_step	(pymea-
ch_B (pymeasure.instruments.advantest.advantestR624X.A attribute), 118		property), 528	
change_source_current (pymea- sure.instruments.advantest.advantestR624X.SMU		_sweep_stop	(pymea-
property), 133	Channei	property), 528	T IVI
	channel	* * * ·	(рутеа-
sure.instruments.advantest.advantestR624X.SMU property), 130			
Channel (class in pymeasure.instruments), 105	channel	_up_relative()	(рутеа-
channel (pymeasure.instruments.bkprecision.BKPrecision property), 248	n9130B	sure.instruments.rohdeschwarz.sfm.Si method), 528	FM
channel1 (pymeasure.instruments.srs.SR830 property), 575	Channel	AFG (class in sure.instruments.activetechnologies.A	pymea- WG401x).
channel1_enabled (pymea-		114	<i>"</i>
sure.instruments.toptica.ibeamsmart.IBeamSma	rt channel		713A at-
property), 634		tribute), 310	II. D.M.C.T.C.
channel2 (pymeasure.instruments.srs.SR830 property), 575	channel	s (pymeasure.instruments.keithley.Keit attribute), 383	thleyDMM6500
channel2_enabled (pymea- sure.instruments.toptica.ibeamsmart.IBeamSmar		s_from_rows_columns()	(pymea-
sure.instruments.tobilca.tbeamsmart.1beamsma	Ιl	sure.msirumenis.keimiey.Keimiey2/0	U

method), 365		method), 267	
characterize()		check_errors() (pymeasure.instruments.hp.HP34	!78A
sure.instruments.keithley.Keithley2	2281S	method), 269	
method), 338		check_errors() (pymeasure.instruments.hp.HP4	!37B
charge (pymeasure.instruments.agilent.a	AgilentB2985	method), 311	
property), 209		<pre>check_errors() (pymeasure.instruments.hp.HP66</pre>	532A
charge_range	(pymea-	method), 317	
sure.instruments.agilent.AgilentB2	2985 prop-	<pre>check_errors() (pymeasure.instruments.hp.HP81</pre>	16A
erty), 209		method), 272	
check_done()	(pymea-	check_errors() (pymeasure.instruments.hp.HP86	557B
sure.instruments.hp.hp856Xx.HP8		method), 305	
method), 276		check_errors() (pymeasure.instruments.Instrum	nent
check_errors()	(рутеа-	method), 103	
sure.instruments.advantest.advant			nea-
method), 119	0511102 111.11414	sure.instruments.keithley.Keithley2000	rica
check_errors()	(рутеа-		
sure.instruments.agilent.agilentB1			nea-
method), 186	300.11giiemB1	sure.instruments.keithley.Keithley2182	пец-
check_errors()	(mymag	method), 395	
	(pymea-		
sure.instruments.agilent.agilentBl	300.SM 0		пеа-
method), 190		sure.instruments.keithley.Keithley2200	
check_errors()	(pymea-	method), 332	
sure.instruments.agilent.AgilentES	52/0B		пеа-
method), 213	,	sure.instruments.keithley.Keithley2260B	
check_errors()	(рутеа-	method), 335	
sure.instruments.aimtti.ld400p.LD	0400P		пеа-
method), 217		sure.instruments.keithley.Keithley2281S	
check_errors()	(рутеа-	method), 338	
sure.instruments.anaheimautomat	ion.DPSeriesM		пеа-
method), 225		sure.instruments.keithley.Keithley2306	
check_errors()	(pymea-	method), 341	
sure.instruments.andeenhagerling.	.AH2700A	check_errors() (pyn	nea-
method), 228		sure.instruments.keithley.Keithley2400	
check_errors()	(pymea-	method), 344	
sure.instruments.anritsu.AnritsuM	IS464xB	check_errors() (pyn	nea-
method), 238		sure.instruments.keithley.Keithley2450	
check_errors()	(pymea-	method), 353	
sure.instruments.anritsu.anritsuM			nea-
method), 243		sure.instruments.keithley.Keithley2510	
check_errors() (pymeasure.instrum	ents.Channel	method), 359	
method), 105		check_errors() (pyn	nea-
check_errors()	(рутеа-	sure.instruments.keithley.Keithley2600	rica
sure.instruments.common_base.Co	* *	method), 362	
method), 99	ommonbase	check_errors() (pyn	naa
check_errors()	(mymag		пец-
	(pymea-	sure.instruments.keithley.Keithley2700	
sure.instruments.fwbell.FWBell50	80 method),	method), 365	
257	,	check_errors() (pyn	пеа-
check_errors()	(pymea-	sure.instruments.keithley.Keithley2750	
sure.instruments.generic_types.SC	PIMixin	method), 368	
method), 108	,	check_errors() (pyn	пеа-
check_errors()	(pymea-	sure.instruments.keithley.Keithley6221	
sure.instruments.heidenhain.ND28	87 method),	method), 371	
259		check_errors() (pyn	пеа-
<pre>check_errors() (pymeasure.instruments.</pre>	.hp.HP3437A	sure.instruments.keithley.Keithley6517B	

method), 378		sure.instruments.keithley.Keithley2000	
check_errors()	(pymea-	method), 323	
sure.instruments.keithley.KeithleyDAQ	26510	<pre>check_get_errors()</pre>	(pymea-
method), $400$		sure.instruments.keithley.Keithley2182	
check_errors()	(pymea-	method), 395	
sure.instruments.keysight.Keysight811	160A	<pre>check_get_errors()</pre>	(pymea-
method), 433		sure.instruments.keithley.Keithley2200	
check_errors()	(pymea-	method), 332	
sure.instruments.keysight.KeysightE36	6312A	<pre>check_get_errors()</pre>	(рутеа-
method), 415		sure.instruments.keithley.Keithley2260	B
check_errors()	(pymea-	method), 335	
sure.instruments.keysight.KeysightE36	631A	<pre>check_get_errors()</pre>	(рутеа-
method), 423		sure.instruments.keithley.Keithley2281	S
check_errors()	(pymea-	method), 338	
sure.instruments.lecroy.LeCroyT3DS0	01204	<pre>check_get_errors()</pre>	(pymea-
method), 459		sure.instruments.keithley.Keithley2306	
check_errors()	(pymea-	method), 341	
sure.instruments.proterial.ROD4	method),	<pre>check_get_errors()</pre>	(рутеа-
509		sure.instruments.keithley.Keithley2400	
check_errors()	(pymea-	method), 344	
sure.instruments.signalrecovery.DSP7	7225	<pre>check_get_errors()</pre>	(pymea-
method), 552		sure.instruments.keithley.Keithley2450	
check_errors()	(pymea-	method), 353	
sure.instruments.signalrecovery.DSP7	* *		(рутеа-
method), 559		sure.instruments.keithley.Keithley2510	4.5
check_errors()	(рутеа-	method), 359	
sure.instruments.srs.ldc500series.LDC	C500Series		(рутеа-
method), 568		sure.instruments.keithley.Keithley2600	
check_errors()	(pymea-	method), 363	
sure.instruments.tdk.tdk_gen40_38.TI			(pymea-
method), 590		sure.instruments.keithley.Keithley2700	* *
check_errors()	(рутеа-	method), 365	
sure.instruments.tdk.tdk_gen80_65.TI	OK Gen80		(pymea-
method), 594		sure.instruments.keithley.Keithley2750	* *
check_errors()	(pymea-	method), 368	
sure.instruments.teledyne.TeledyneT32	4 .		(pymea-
method), 600		sure.instruments.keithley.Keithley6221	
	(рутеа-	method), 371	
sure.instruments.texio.TexioPSW360L		<pre>check_get_errors()</pre>	(рутеа-
method), 622		sure.instruments.keithley.Keithley6517	B
check_get_errors()	(pymea-	method), 379	
sure.instruments.andeenhagerling.AH		<pre>check_get_errors()</pre>	(pymea-
method), 228		sure.instruments.keithley.KeithleyDAQ	
<pre>check_get_errors() (pymeasure.instruments</pre>	s.Channel	method), 400	
method), 105		<pre>check_get_errors()</pre>	(pymea-
check_get_errors()	(рутеа-	sure.instruments.keysight.Keysight8110	
sure.instruments.common_base.Comn	nonBase	method), 433	
method), 99		<pre>check_get_errors()</pre>	(рутеа-
check_get_errors()	(pymea-	sure.instruments.keysight.KeysightE36	
		method), 415	
257	**	<pre>check_get_errors()</pre>	(pymea-
check_get_errors()	(рутеа-	sure.instruments.keysight.KeysightE36	
sure.instruments.Instrument method),	103	method), 423	
check_get_errors()	(pymea-	<pre>check_get_errors()</pre>	(pymea-

sure.instruments.lecroy.LeCroyT3DS0 method), 460	01204	<pre>check_set_errors()           sure.instruments.hcp.TC038 method),</pre>	(pymea-
	(m) m a a	check_set_errors()	
check_get_errors()	(pymea-		(pymea-
sure.instruments.proterial.ROD4	method),	sure.instruments.hcp.TC038D method)	
509	(	check_set_errors()	(pymea-
<pre>check_get_errors()</pre>	(pymea-	sure.instruments.inficon.sqm160.SQM	100
sure.instruments.signalrecovery.DSP7	/223	method), 320	,
method), 552	,	<pre>check_set_errors()</pre>	(pymea-
<pre>check_get_errors()</pre>	(pymea-	sure.instruments.Instrument method),	
sure.instruments.signalrecovery.DSP7	/203	<pre>check_set_errors()</pre>	(рутеа-
method), 559	,	sure.instruments.ipgphotonics.yar.YAR	
<pre>check_get_errors()</pre>	(pymea-		,
sure.instruments.tdk.tdk_gen40_38.TI	OK_Gen40_		(рутеа-
method), 590	,	sure.instruments.keithley.Keithley2000	!
<pre>check_get_errors()</pre>	(pymea-	method), 324	,
sure.instruments.tdk.tdk_gen80_65.TI	OK_Gen80_		(рутеа-
method), 594	,	sure.instruments.keithley.Keithley2182	
<pre>check_get_errors()</pre>	(pymea-	method), 395	,
sure.instruments.teledyne.TeledyneT3.	AFG		(рутеа-
method), 600	,	sure.instruments.keithley.Keithley2200	1
<pre>check_get_errors()</pre>	(pymea-	method), 333	,
sure.instruments.texio.TexioPSW360L	.30	<pre>check_set_errors()</pre>	(рутеа-
method), 622		sure.instruments.keithley.Keithley2260	B
<pre>check_get_estimates_signature()</pre>		method), 335	
sure.display.widgets.estimator_widget	t.Estimator\		(рутеа-
method), 82		sure.instruments.keithley.Keithley2281	S
<pre>check_idle()</pre>	(pymea-	method), 338	
sure.instruments.agilent.agilentB1500	).AgilentB1.		(рутеа-
method), 186		sure.instruments.keithley.Keithley2306	
check_parameters()	(pymea-	method), 341	
sure.experiment.procedure.Procedure	method),		(рутеа-
65		sure.instruments.keithley.Keithley2400	)
<pre>check_selftest_errors()</pre>	(рутеа-	method), 345	
sure.instruments.hp.HP6632A method		<pre>check_set_errors()</pre>	(рутеа-
check_set_errors()	(pymea-	sure.instruments.keithley.Keithley2450	)
sure.instruments.aculight.argos.Argos	8	method), 353	,
method), 116		<pre>check_set_errors()</pre>	(рутеа-
<pre>check_set_errors()</pre>	(pymea-		)
sure.instruments.ametek.Ametek7270	method),	method), 359	
220		<pre>check_set_errors()</pre>	(рутеа-
check_set_errors()	(рутеа-	sure.instruments.keithley.Keithley2600	)
sure.instruments.andeenhagerling.AH	12700A	method), 363	
method), 228		<pre>check_set_errors()</pre>	(рутеа-
<pre>check_set_errors()</pre>	(pymea-	sure.instruments.keithley.Keithley2700	1
sure.instruments.attocube.anc300.AN	C300Contro		
method), 246		<pre>check_set_errors()</pre>	(рутеа-
check_set_errors() (pymeasure.instrument.	s.Channel	sure.instruments.keithley.Keithley2750	1
method), 106	,	method), 369	
check_set_errors()	(pymea-	<pre>check_set_errors()</pre>	(рутеа-
sure.instruments.common_base.Comm	nonBase	sure.instruments.keithley.Keithley4200	'
method), 99	,	method), 405	,
check_set_errors()	(pymea-	<pre>check_set_errors()</pre>	(рутеа-
sure.instruments.fwbell.FWBell5080	method),	sure.instruments.keithley.Keithley6221	
257		method), 372	

check_set_errors() sure.instruments.keithley.Keithley651	(pymea- 7B	check_s	et_errors() sure.instruments.tdk.tdk_gen80_65.Ti	(pymea- DK_Gen80_65
method), 379 check_set_errors() sure.instruments.keithley.KeithleyDAQ	(pymea- Q6510	check_s	method), 594 et_errors() sure.instruments.teledyne.TeledyneT3	(pymea- PAFG
method), 400 check_set_errors() sure.instruments.keysight.Keysight81	(pymea- 160A	check_s	method), 600 et_errors() sure.instruments.texio.TexioPSW3601	(pymea- L30
method), 433 check_set_errors() sure.instruments.keysight.KeysightE3 method), 416	(pymea- 6312A	check_s	method), 623 et_errors() sure.instruments.thyracont.smartline_ method), 628	(pymea- _v1.SmartlineV1
check_set_errors()  sure.instruments.keysight.KeysightE3  method), 424	(pymea- 631A	check_s	et_errors() sure.instruments.thyracont.smartline_ method), 630	(pymea- _v2.SmartlineV2
check_set_errors()  sure.instruments.lecroy.LeCroyT3DS0  method), 460	(pymea- 01204	check_s	et_errors() sure.instruments.toptica.ibeamsmart. method), 634	(pymea- IBeamSmart
check_set_errors() sure.instruments.mksinst.mksinst.MKs method), 471	(pymea- SInstrument		et_errors() sure.instruments.velleman.VellemanK method), 636	(pymea- (8090
check_set_errors() sure.instruments.novanta.Fpu60 492	(pymea- method),	check_s	top() sure.display.windows.plotter_window method), 94	(pymea- PlotterWindow.
check_set_errors() sure.instruments.oxfordinstruments.m method), 503			emperature_stability() sure.instruments.keithley.Keithley251 method), 360	(pymea- 0
check_set_errors()	(nvmea-	chock +		
sure. instruments. ox for dinstruments. m			<pre>emperature_stability() eSunsonstruments.srs.ldc500series.LD method), 571</pre>	(pymea- C500SeriesTEC
sure.instruments.oxfordinstruments.m method), 502 check_set_errors() sure.instruments.proterial.ROD4		emperature checksu	e <b>Suns</b> anstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellman static method), 565	C500SeriesTEC hv.SpellmanXRV
sure.instruments.oxfordinstruments.m method), 502 check_set_errors() sure.instruments.proterial.ROD4 509 check_set_errors() sure.instruments.ptw.ptwDIAMENTO	ercuryitc.Te (pymea- method), (pymea-	emperature checksu	e Suns anstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellman. static method), 565 (pymeasure.experiment.parameters.Li property), 68 (pymeasure.display.manager.Bas	C500SeriesTEC hv.SpellmanXRV istParameter
sure.instruments.oxfordinstruments.m method), 502 check_set_errors() sure.instruments.proterial.ROD4 509 check_set_errors() sure.instruments.ptw.ptwDIAMENTO method), 512 check_set_errors() sure.instruments.ptw.ptwUNIDOS	ercuryitc.Te (pymea- method), (pymea-	checksu choices clear() clear()	esunsanstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellman. static method), 565 (pymeasure.experiment.parameters.Li property), 68 (pymeasure.display.manager.Bas method), 79 (pymeasure.instruments.andeenhager. method), 228	C500SeriesTEC hv.SpellmanXRV istParameter eManager ling.AH2700A
sure.instruments.oxfordinstruments.m method), 502 check_set_errors() sure.instruments.proterial.ROD4 509 check_set_errors() sure.instruments.ptw.ptwDIAMENTO method), 512 check_set_errors() sure.instruments.ptw.ptwUNIDOS 514 check_set_errors() sure.instruments.signalrecovery.DSP	(pymea- method), (pymea- R (pymea- method), (pymea- method), (pymea-	checksu choices clear() clear()	esunsanstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellmanstatic method), 565 (pymeasure.experiment.parameters.Lt.property), 68	C500SeriesTEC hv.SpellmanXRV istParameter eManager ling.AH2700A WBell5080
sure.instruments.oxfordinstruments.m method), 502  check_set_errors() sure.instruments.proterial.ROD4 509  check_set_errors() sure.instruments.ptw.ptwDIAMENTO method), 512  check_set_errors() sure.instruments.ptw.ptwUNIDOS 514  check_set_errors() sure.instruments.signalrecovery.DSP' method), 552  check_set_errors() sure.instruments.signalrecovery.DSP' sure.instruments.signalrecovery.DSP'	ercuryitc.Te  (pymea- method),  (pymea- R  (pymea- method),  (pymea- 7225  (pymea-	checksu choices clear() clear() clear() clear() clear()	esunsinstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellman. static method), 565 (pymeasure.experiment.parameters.Li property), 68	C500SeriesTEC hv.SpellmanXRV istParameter eManager ling.AH2700A WBell5080 s.SCPIMixin method),
sure.instruments.oxfordinstruments.m method), 502  check_set_errors() sure.instruments.proterial.ROD4 509  check_set_errors() sure.instruments.ptw.ptwDIAMENTO method), 512  check_set_errors() sure.instruments.ptw.ptwUNIDOS 514  check_set_errors() sure.instruments.signalrecovery.DSP2 method), 552  check_set_errors()	ercuryitc.Te (pymea- method), (pymea- R (pymea- method), (pymea- 7225 (pymea- 7265 (pymea-	checksuch	esunsanstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellman. static method), 565 (pymeasure.experiment.parameters.Li property), 68	C500SeriesTEC hv.SpellmanXRV istParameter eManager ling.AH2700A WBell5080 s.SCPIMixin method), method),
sure.instruments.oxfordinstruments.m method), 502  check_set_errors() sure.instruments.proterial.ROD4 509  check_set_errors() sure.instruments.ptw.ptwDIAMENTO method), 512  check_set_errors() sure.instruments.ptw.ptwUNIDOS 514  check_set_errors() sure.instruments.signalrecovery.DSP method), 552  check_set_errors() sure.instruments.signalrecovery.DSP method), 559 check_set_errors()	ercuryitc.Te  (pymea- method),  (pymea- R  (pymea- method),  (pymea- 7225  (pymea- 7265  (pymea- 72KRV  (pymea-	checksuch	esunsinstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellmanstatic method), 565 (pymeasure.experiment.parameters.Laproperty), 68	C500SeriesTEC hv.SpellmanXRV istParameter eManager ling.AH2700A WBell5080 s.SCPIMixin method), method), method),
sure.instruments.oxfordinstruments.m method), 502  check_set_errors() sure.instruments.proterial.ROD4 509  check_set_errors() sure.instruments.ptw.ptwDIAMENTO method), 512  check_set_errors() sure.instruments.ptw.ptwUNIDOS 514  check_set_errors() sure.instruments.signalrecovery.DSP: method), 552  check_set_errors() sure.instruments.signalrecovery.DSP: method), 559  check_set_errors() sure.instruments.spellmanhv.Spellman method), 564  check_set_errors() sure.instruments.srs.ldc500series.LDo method), 568  check_set_errors()	ercuryitc.Te  (pymea- method),  (pymea- R  (pymea- method),  (pymea- 7225  (pymea- 7265  (pymea- 7265)	checksuchecksuchoices clear() clear() clear() clear() clear() clear() clear() clear() clear()	esunsanstruments.srs.ldc500series.LD method), 571 m() (pymeasure.instruments.spellman. static method), 565 (pymeasure.experiment.parameters.Li property), 68	C500SeriesTEC hv.SpellmanXRV istParameter eManager ling.AH2700A WBell5080 s.SCPIMixin method), method), method), cs.yar.YAR ithley2000

	(pymeasure.instruments.keithley.Keithley2260B method), 335	<pre>clear_average_count()     sure.instruments.anritsu.anritsuMS464</pre>	(pymea- 4xB.MeasurementChannel
clear()	(pymeasure.instruments.keithley.Keithley2281S method), 338	<pre>method), 243 clear_buffer()</pre>	(рутеа-
clear()	(pymeasure.instruments.keithley.Keithley2306 method), 341	sure.instruments.agilent.agilentB1500 method), 186	
clear()	(pymeasure.instruments.keithley.Keithley2400 method), 345	<pre>clear_display()</pre>	(pymea- method).
clear()	(pymeasure.instruments.keithley.Keithley2450 method), 353	176 clear_display()	(pymea-
clear()	(pymeasure.instruments.keithley.Keithley2510 method), 360	sure.instruments.keysight.Keysight811 method), 433	• •
	(pymeasure.instruments.keithley.Keithley2600 method), 363	clear_errors() sure.instruments.newport.ESP300	(pymea- method),
clear()	(pymeasure.instruments.keithley.Keithley2700 method), 365	476 clear_measurement_buffer()	(рутеа-
clear()	(pymeasure.instruments.keithley.Keithley2750 method), 369	sure.instruments.advantest.advantestR $method), 127$	624X.SMUChannel
clear()	(pymeasure.instruments.keithley.Keithley4200 method), 405	<pre>clear_overload() (pymeasure.instruments.s     method), 573</pre>	ers.SR570
clear()	(pymeasure.instruments.keithley.Keithley6221 method), 372	<pre>clear_plot()      sure.experiment.experiment.Experiment</pre>	(pymea- nt
clear()	(pymeasure.instruments.keithley.Keithley6517B method), 379	<pre>method), 63 clear_ramp_set()</pre>	(pymea-
clear()	(pymeasure.instruments.keithley.KeithleyDAQ651 method), 401	- · · · · · · · · · · · · · · · · · · ·	• •
clear()	(pymeasure.instruments.keysight.Keysight81160A method), 433		(pymea- 90
clear()	(pymeasure.instruments.keysight.KeysightE36312 method), 416		(pymea-
clear()	(pymeasure.instruments.keysight.KeysightE3631A method), 424		
clear()	(pymeasure.instruments.lecroy.LeCroyT3DSO120 method), 460		(pymea- OX1102G
clear()		method), 409 clear_status_register()	(рутеа-
clear()		sure.instruments.advantest.advantestR method), 119	
clear()	(pymeasure.instruments.signalrecovery.DSP7225 method), 552		(pymea- AgilentB1500
clear()	(pymeasure.instruments.signalrecovery.DSP7265 method), 559		(рутеа-
clear()		sure.instruments.kuhneelectronic.Kusg method), 442	
clear()	(pymeasure.instruments.tdk.tdk_gen40_38.TDK_0 method), 590		(pymea- Vidaet
clear()	$(pymeasure.instruments.tdk.tdk\_gen80\_65.TDK\_c$	Gen80_65 method), 85	
clear()	method), 594 (pymeasure.instruments.teledyne.TeledyneT3AFG		(pymea- idget
clear()	* *	method), 87 clear_widget()	(pymea-
clear()	method), 614 (pymeasure.instruments.texio.TexioPSW360L30	sure.display.widgets.table_widget.Tabl method), 90	
	method), 623	<pre>clear_write_trace()</pre>	(pymea-

	sure.instruments.hp.hp856Xx.HP856X.method), 288	x	attribute), 632 coldcathode_pressure	(рутеа-
cli_args (pymeasure.experiment.parameters.Parameter		sure.instruments.mksinst.mks974b.MK		
	property), 70	47	property), 475	(
	(pymeasure.adapters.Adapter method),		<pre>collapse_channel_string()</pre>	(pymea-
	(pymeasure.adapters.FakeAdapter meta		sure.instruments.ni.virtualbench.Virtu	аіВепсп
close()	(pymeasure.adapters.PrologixAdapter	method),	method), 490	7. 7
1 0	55	1 1 50	colormap() (pymeasure.display.curves.Rest	ultsImage
	(pymeasure.adapters.SerialAdapter me		method), 77	(
	(pymeasure.adapters.VISAAdapter met		columnCount()	(pymea-
	(pymeasure.instruments.keithley.Keith method), 369		sure.display.widgets.sequencer_widge method), 86	t.Sequencer1reeModel
close()	(pymeasure.instruments.keithley.Keithl	eyDMM65		(рутеа-
	method), 385		sure.display.widgets.table_widget.Pan	dasModelBase
close()	(pymeasure.instruments.keysight.Keysig	ghtN7776C		
	method), 411		combined_pressure1	(рутеа-
close_c	hannel()	(pymea-	sure.instruments.mksinst.mks937b.MK	<i>S937B</i>
	sure.instruments.keithley.KeithleyDAQ	6510	property), 474	
	method), 401		combined_pressure2	(рутеа-
close_c	hannels()	(pymea-	sure.instruments.mksinst.mks937b.MK	<i>S937B</i>
	sure.instruments.keithley.KeithleyDAQ	6510	property), 474	
	method), 401		ComboBoxDelegate (class in	рутеа-
close_r	ows_to_columns()	(pymea-	sure.display.widgets.sequencer_widge	<i>t</i> ),
	sure.instruments.keithley.Keithley 2700		85	
	method), 365		COMMAND_COMPLETE	(рутеа-
closed_	channels	(pymea-	sure.instruments.hp.hp856Xx.StatusRe	egister
	sure. instruments. keithley. Keithley 2700	prop-	attribute), 303	
	erty), 365		command_set (pymeasure.instruments.keithley.l	KeithleyDMM6500
closed_	channels	(pymea-	property), 385	
	sure.instruments.keithley.Keithley2750	prop-	CommonBase (class in	рутеа-
	erty), 369		sure.instruments.common_base), 97	
CMU_MEA	SUREMENT	(pymea-		ı рутеа-
	sure.instruments.agilent.agilentB1500. attribute), 197	WaitTimeT		рутеа-
CNT91 (c.	lass in pymeasure.instruments.pendulur	n.cnt91),	sure.instruments.common_base), 97	
	507		CommonBase.MultiChannelCreator (class in	п рутеа-
coder_a		(pymea-	sure.instruments.common_base), 98	
	sure.instruments.rohdeschwarz.sfm. SF	M	complement_enabled	(рутеа-
	method), 528		sure.instruments.hp.HP8116A p	property),
coder_i	d_frequency	(pymea-	272	
	sure.instruments.rohdeschwarz.sfm.SF	M	$\verb complete  (pymeasure.instruments.andeen hager)  \\$	ling.AH2700A
	property), 528		property), 228	
coder_m	odulation_degree	(pymea-	complete (pymeasure.instruments.fwbell.FW	Bell5080
	sure.instruments.rohdeschwarz.sfm.SF	M	property), 258	
	property), 529		<pre>complete(pymeasure.instruments.generic_type</pre>	s.SCPIMixin
coder_p	ilot_deviation	(pymea-	property), 108	
	sure.instruments.rohdeschwarz.sfm.SF	M	complete (pymeasure.instruments.hp.HP8116	A prop-
	property), 529		erty), 272	
coder_p	ilot_frequency	(рутеа-	complete (pymeasure.instruments.Instrument p	property),
	sure.instruments.rohdeschwarz.sfm.SF	M	104	
	property), 529		complete (pymeasure.instruments.keithley.Keit	hley2000
coilcon	st (pymeasure.instruments.ami.AMI43	30 prop-	property), 324	
	erty), 223		complete (pymeasure.instruments.keithley.Keit	hley2182
coldcat	hode (pymeasure.instruments.thyracom	.smartline	_v2.VSM property), 395	

complete (pymeasure.instruments.keithley.Keithley2200 property), 333	COMPLIANCE_AND_FORCE_SIDE sure.instruments.agilent.agilentB1500	(pymea- MeasOpMode
complete (pymeasure.instruments.keithley.Keithley2260B	attribute), 196	,
property), 335	<del>-</del>	(pymea-
complete (pymeasure.instruments.keithley.Keithley2281S property), 338	sure.instruments.keithley.Keithley2400 erty), 345	prop-
complete (pymeasure.instruments.keithley.Keithley2306	compliance_current	(pymea-
property), 342	sure.instruments.keithley.Keithley2450	prop-
complete (pymeasure.instruments.keithley.Keithley2400	erty), 353	
property), 345	compliance_current	(pymea-
complete (pymeasure.instruments.keithley.Keithley2450 property), 353	sure.instruments.yokogawa.Yokogawa7 property), 637	651
complete (pymeasure.instruments.keithley.Keithley2510	COMPLIANCE_SIDE	(рутеа-
property), 360	sure.instruments.agilent.agilentB1500	MeasOpMode
complete (pymeasure.instruments.keithley.Keithley2600	attribute), 195	
property), 363	compliance_voltage	(рутеа-
complete (pymeasure.instruments.keithley.Keithley2700	sure.instruments.keithley.Keithley2400	prop-
property), 365	erty), 345	
complete (pymeasure.instruments.keithley.Keithley2750	compliance_voltage	(pymea-
property), 369	sure.instruments.keithley.Keithley2450	prop-
complete (pymeasure.instruments.keithley.Keithley6221	erty), 353	
property), 372	compliance_voltage	(рутеа-
$\verb complete  (pymeasure. instruments. keithley. Keithley 6517B $	sure.instruments.yokogawa.Yokogawa7	651
property), 379	property), 637	
complete (pymeasure.instruments.keithley.KeithleyDAQ65		pymea-
property), 401	sure.instruments.agilent.agilentB1500)	
complete (pymeasure.instruments.keysight.Keysight81160.		(pymea-
property), 433	sure.instruments.temptronic.ATSBase	prop-
complete (pymeasure.instruments.keysight.KeysightE3631		(
property), 416		(рутеа-
complete (pymeasure.instruments.keysight.KeysightE3631 property), 424	property), 407	
complete (pymeasure.instruments.lecroy.LeCroyT3DSO12 property), 460	<b>Config</b> (pymeasure.instruments.andeenhagerling property), 227	g.AH2500A
<pre>complete (pymeasure.instruments.proterial.ROD4 prop-</pre>	config (pymeasure.instruments.andeenhagerling	g.AH2700A
erty), 510	property), 228	
$\verb complete  (pymeasure. instruments. signal recovery. DSP722. See the control of the control o$	5config_amplitude_modulation()	(рутеа-
property), 552	sure.instruments.agilent.Agilent8257D	
$\verb complete  (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. DSP726. Section 1)   Complete (pymeasure. instruments. signal recovery. Section 1)   Complete (pymeasure. inst$	5 method), 149	
property), 559	<pre>config_buffer()</pre>	(рутеа-
${\tt complete} \ (pymeasure.instruments.tdk.tdk\_gen40\_38.TDK\_$	_Gen40_3&ure.instruments.keithley.Keithley2000	
property), 590	method), 324	
$\verb complete  (pymeasure.instruments.tdk.tdk\_gen80\_65.TDK\_ $	_ <i>Com&amp;D</i> g <u>6</u> buffer()	(рутеа-
property), 594	sure.instruments.keithley.Keithley2182	
$\verb complete  (pymeasure. instruments. teledyne. Teledyne T3AF (example))   (pymeasure. instruments. teledyne. Teledyne T3AF (example))   (pymeasure. instruments. instruments$	G method), 396	
property), 600	<pre>config_buffer()</pre>	(pymea-
$\verb complete  (pymeasure.instruments.texio.TexioPSW360L30 $	sure.instruments.keithley.Keithley2400	
property), 623	method), 345	
$\verb compliance   (pymeasure.instruments.agilent.agilent4156.S)   $	Monfig_buffer()	(pymea-
property), 169	sure.instruments.keithley.Keithley2450	
$\verb compliance   (pymeasure.instruments.agilent.agilent4156.V )  $	YARD method), 353	
property), 170	<pre>config_buffer()</pre>	(pymea-
$\verb compliance   (pymeasure.instruments.agilent.agilent4156.V )  $	YARX sure.instruments.keithley.Keithley2700	
property), 171	method), 366	

<pre>config_buffer()</pre>	(pymea-	method), 162	
sure.instruments.keithley.Keithley622	!	<pre>configure_current()</pre>	(pymea-
method), 372		sure.instruments.agilent.Agilent34450	OA.
<pre>config_buffer()</pre>	(рутеа-	method), 162	
sure.instruments.keithley.Keithley6517	<sup>7</sup> B	<pre>configure_current_output()</pre>	
method), 379	,	sure.instruments.ni.virtualbench.Virtu	ialBench.PowerSupply
config_buffer()	(pymea-	method), 489	(
sure.instruments.keithley.KeithleyDAQ	<i>j</i> 6510	configure_dc_current()	(pymea-
<pre>method), 401 config_low_freq_out()</pre>	(mymag	sure.instruments.ni.virtualbench.Virtu method), 480	iaiBench.DigitaiMuitimete
sure.instruments.agilent.Agilent8257L	(pymea-	configure_dc_voltage()	(рутеа-
method), 149	,	sure.instruments.ni.virtualbench.Virtu	4.5
config_pulse_modulation()	(pymea-	method), 480	andenen.Dignannunnene
sure.instruments.agilent.Agilent8257L		configure_diode()	(рутеа-
method), 149		sure.instruments.agilent.Agilent34450	
<pre>config_step_sweep()</pre>	(pymea-	method), 162	
sure.instruments.agilent.Agilent8257L			(pymea-
method), 150		sure.instruments.agilent.Agilent34450	
configuration	(pymea-	method), 162	
sure.instruments.spellmanhv.Spellmannuts.s	XRV	configure_frequency_array_measuremen	t()
property), 565		(pymeasure.instruments.pendulum.cn	t91.CNT91
configure() (pymeasure.instruments.agilent.a	gilent4156		
method), 167		<pre>configure_immediate_trigger()</pre>	
configure() (pymeasure.instruments.temptron	ic.ATSBas		alBench.MixedSignalOsci
method), 614	,	method), 486	
configure_ac_current()		configure_measurement()	(pymea-
	аівепсп.Д	igitalMultimuterinstruments.ni.virtualbench.Virtu	iaiBench.DigitaiMuitimete
method), 480 configure_alarm()	(mymag	<pre>method), 480 configure_relay()</pre>	(pwm ag
sure.instruments.lakeshore.LakeShore		sure.instruments.lakeshore.LakeShore	(pymea- 5211
method), 445	211	method), 445	.211
	(pymea-	configure_resistance()	(рутеа-
		ixedSignal <b>Qurèllastrapn</b> ents.agilent.Agilent34450	
method), 485		method), 162	
**	ics()	<pre>configure_standard_waveform()</pre>	(рутеа-
		ench.Mixed <b>SignalOsaihlesstxopi</b> evirtualbench.Virtu	
method), 485		method), 482	
<pre>configure_analog_edge_trigger()</pre>	(pymea-	<pre>configure_temperature()</pre>	(pymea-
sure. instruments. ni. virtual bench. Virtual bench. Tittuta titta sure tit	alBench.M	ixedSignal <b>@wellnst:apn</b> ents.agilent.Agilent34450	OA.
method), 485		method), 163	
configure_analog_pulse_width_trigger(		<pre>configure_timer()</pre>	(рутеа-
- · · · · · · · · · · · · · · · · · · ·	ch.VirtualB	ench.Mixed <b>SignidOsuitlesstx</b> opiglenttechnologies.s	iglent_spdbase.SPDChann
method), 485	,	method), 546	/
<pre>configure_arbitrary_waveform()</pre>		<pre>configure_timing()</pre>	(pymea-
	аівепсп. Гі	unctionGen <b>suatoin</b> struments.ni.virtualbench.Virtu	iaiBench.MixeaSignaiOsci
method), 482 configure_arbitrary_waveform_gain_and	l offco+(	method), 486	(mayor a a
		confrigure_trigger_deray() ench.Functi <b>son&amp;instratne</b> nts.ni.virtualbench.Virtu	(pymea- yalRanch MiyadSianalOsci
method), 482	н. у н нааг	method), 486	andenen.mixeasignaiosei
configure_capacitance()	(pymea-	configure_voltage()	(рутеа-
sure.instruments.agilent.Agilent34450	4.0	sure.instruments.agilent.Agilent34450	
method), 162		method), 163	
configure_continuity()	(рутеа-	<pre>configure_voltage_output()</pre>	(pymea-
sure.instruments.agilent.Agilent34450	A	sure.instruments.ni.virtualbench.Virtu	* *

method), 489		control	_loop_heater_percent	(рутеа-
ConsoleArgumentParser (class in	рутеа-		sure.instruments.oxfordinstruments.me	
sure.display.console), 75	руппес		property), 503	reur yue. 1emperaum esemse
ConsoleBrowserItem (class in	nvmea-	control.	· ·	(рутеа-
sure.display.console), 75	рутеа-	CONCLOS	_100p_1 sure.instruments.oxfordinstruments.me	4.5
	(222220 0 0			rcuryuc.1emperaturesenso
) — — <u>·</u>	(pymea-		property), 503	(2000) 0.00
sure.instruments.ptw.ptwDIAMENTOR	x prop-	control.	=	(pymea-
erty), 512	,		sure.instruments.oxfordinstruments.me	rcuryiic.1emperatureSenso
constant_value	(pymea-		property), 503	,
sure.instruments.agilent.agilent4156.S	MU	control.	<del>-</del>	(pymea-
property), 169			sure.instruments.oxfordinstruments.me	rcuryitc.TemperatureSenso
constant_value	(pymea-		property), 503	
sure.instruments.agilent.agilent4156.V	'SU	control.		(pymea-
property), 171			sure.instruments.ox for dinstruments.me	rcury itc. Temperature Senso
contact_current_1	(pymea-		property), 503	
sure.instruments.razorbill.razorbillRP	100	control	_loop_ramp_rate	(pymea-
property), 521			sure.instruments.oxfordinstruments.me	rcuryitc.TemperatureSenso
contact_current_2	(рутеа-		property), 503	
sure.instruments.razorbill.razorbillRP				(pymea-
property), 521			sure.instruments.oxfordinstruments.me	
contact_voltage_1	(pymea-		property), 503	1
sure.instruments.razorbill.razorbillRP		control.		(рутеа-
property), 521			sure.instruments.rohdeschwarz.hmp.Hi	
contact_voltage_2	(рутеа-		property), 542	,11 70 70
sure.instruments.razorbill.razorbillRP.		control.		IP81164
property), 521	100	CONCLOS	property), 272	11 0110/1
contextMenuEvent()	(рутеа-	control		(mymag
				(pymea- 5120-10
sure.display.widgets.dock_widget.Dock	awiagei		sure.instruments.oxfordinstruments.IPS	3120_10
method), 88	,		property), 499	,
<pre>continue_single_sweep()</pre>	(pymea-			(pymea-
sure.instruments.rohdeschwarz.fsseries	s.FSSeries		sure.instruments.oxfordinstruments.ITC	2503
method), 540			property), 495	
${\tt continuity} \ (py measure. instruments. a gilent. A$	ilent34450.	Acontrol		(pymea-
property), 163			sure.instruments.attocube.anc 300.ANC	C300Controller
${\tt continuous}\ (py measure. instruments. pendulum.$	cnt91.CNT		property), 246	
property), 508		convers	ion_loss	(pymea-
continuous_sweep_enabled	(pymea-		sure.instruments.hp.HP8561B p	roperty),
sure.instruments.rohdeschwarz.fsseries	s.FSSeries		298	
property), 540		convert	() (pymeasure.experiment.parameters.H	BooleanParameter
<pre>control() (pymeasure.instruments.common_ba</pre>	ise.Commo	nBase	method), 67	
static method), 99		convert	() (pymeasure.experiment.parameters.I	FloatParameter
control() (pymeasure.instruments.fakes.FakeIn	nstrument		method), 67	
static method), 107		convert	() (pymeasure.experiment.parameters.I	ntegerParameter
control() (pymeasure.instruments.keysight.Ke	vsight8116		method), 68	
static method), 433	/~~0		() (pymeasure.experiment.parameters.I	istParameter
control() (pymeasure.instruments.keysight.Ke	vsiohtF363		method), 68	
static method), 416	ysigniEsos		() (pymeasure.experiment.parameters.I	Parameter
control() (pymeasure.instruments.keysight.Ke	veiahtF363		method), 70	arameter
	ysigniE303			Dhysical Darameter
static method), 424	ONT2DCA		() (pymeasure.experiment.parameters.I	nysican arameier
control() (pymeasure.instruments.lecroy.LeCr	oyi s <b>DS</b> OI		method), 70	Vactor Dane t
static method), 460	(	convert	() (pymeasure.experiment.parameters.V	eciorparameter
control_loop_D	(pymea-		method), 71	(
sure.instruments.oxfordinstruments.me	rcuryitc.Te	napanante		(pymea-
property), 502			sure.instruments.ni.virtual bench. Virtual bench.	ılBench

me	thod), 490			sure.instruments.rohdeschwarz.fsseries	.FSSeries
convert_va	lues_to_datetime()	(pymea-		method), 540	
sur	e.instruments.ni.virtualbench.Virtua	alBench	createE		(рутеа-
	thod), 490			sure.display.widgets.sequencer_widget.	ComboBoxDelegate
convert_va	lues_to_timestamp()	(рутеа-		method), 85	
sur	e.instruments.ni.virtualbench.Virtu	alBench	createE	**	(рутеа-
	thod), 491			sure.display.widgets.sequencer_widget.	LineEditDelegate
	e_setup_file	(рутеа-		method), 85	
	re.instruments.temptronic.ATSBase	prop-		<pre>irs (class in pymeasure.display.curves)</pre>	, 76
	y), 614		crystal		(pymea-
copy_data_		(рутеа-		sure. instruments. in ficon. sqm 160. Senso	rChannel
	e.instruments.anritsu.AnritsuMS464	4xB		property), 321	
	thod), 238		CSVForm	atter (class in pymeasure.experiment	results),
copy_trace		(рутеа-		72	
		ies.AQ6370	O <b>sunir</b> ent	(pymeasure.instruments.agilent.Agilen	t34450A
me	thod), 640			property), 163	
COR	(class in	рутеа-	CURRENT	(pymeasure.instruments.agilent.agilent	B1500.MeasOpMode
	re.instruments.advantest.advantestRe	624X),		attribute), 195	
141				(pymeasure.instruments.agilent.Agile	ntB2981
	(pymeasure.instruments.agilent.Ag	ilent4284A		property), 204	
	ribute), 198		current	(pymeasure.instruments.agilent.agilent	E5270B.SMUChannel
correction		(pymea-		property), 214	
		roperty),	current	(py measure. instruments. aimtti. aimtti PL	PLChannel
30				property), 216	
correction		(рутеа-	current	(pymeasure.instruments.aimtti.ld400p.	LD400P
	e.instruments.ptw.ptwDIAMENTOR	R prop-		property), 217	
	y), 512		current	(pymeasure.instruments.aja.DCXS p.	roperty),
_	ymeasure.instruments.hp.hp856Xx.l	HP856Xx		219	rp 0120p
	ribute), 275			(pymeasure.instruments.bkprecision.Bk	Precision9130B
_	ymeasure.instruments.siglenttechno	logies.sigle			11.0500
	operty), 549	D.C.D.722		(pymeasure.instruments.danfysik.Danf	vs1k8500
_	ymeasure.instruments.signalrecover	ry.DSP/22		property), 249	G147045D
	operty), 553	D.C.D.73.6		(pymeasure.instruments.deltaelektronik	a.SM/045D
_	ymeasure.instruments.signalrecover	ry.DSP/26		property), 252	JIDD 120256
	operty), 559	, ,,,		(pymeasure.instruments.eurotest.Eurote	stHPP120256
	ymeasure.instruments.teledyne.teled	dyne_oscill	_		
	operty), 607	,		(pymeasure.instruments.hp.HP6632A	prop-
coupling_e		(pymea-		erty), 318	174 D
	, , ,	00A.Keysig		Chapymeeasure.instruments.ipgphotonics.	yar.YAR
	operty), 440			property), 322	1 2000
CouplingMo	*	рутеа-	current	(pymeasure.instruments.keithley.Keith	uey2000
	re.instruments.hp.hp856Xx), 301	(		property), 324	2200 DCCl1
create_cha		(pymea-	current	(pymeasure.instruments.keithley.keithle	y2200.PSCnannei
	re.instruments.rohdeschwarz.fsseries	S.FSW		property), 334	2260D
	thod), 542	(	current	(pymeasure.instruments.keithley.Keithle	еу2200В
create_dir	-	(pymea-		property), 335	.12400
	e.instruments.anritsu.AnritsuMS464	4XB	current	(pymeasure.instruments.keithley.Keith	uey2400
	thod), 238	(		property), 345	.12450
	_trace_window()	(pymea-	current	(pymeasure.instruments.keithley.Keith	uey2430
	re.instruments.hp.hp856Xx.HP856X thod)	л	CHENON+	property), 353	alay 2510
me create_fil	thod), 285 ename() (in module	nymea	current	(pymeasure.instruments.keithley.Keith property), 360	ue y 2 3 1 0
	re.experiment.experiment), 64	рутеа-	Curren+	(pymeasure.instruments.keithley.keithle	v4200 SMII
create_mar		(рутеа-	CULTEIL	property), 405	y 1200.01110
CI CU CC_IIIAI	( <i>)</i>	pymen		property), 100	

current (pymeasure.instruments.keithley.Keithley6517B property), 379	current	_ac_digits sure.instruments.keithley.KeithleyDMM	(pymea- 16500
current (pymeasure.instruments.keithley.KeithleyDAQ65)	10	property), 385	10300
property), 401			(pymea-
current (pymeasure.instruments.keithley.KeithleyDMM65		sure.instruments.keithley.Keithley2000	
property), 385	.00	erty), 324	ргор
current (pymeasure.instruments.kepco.KepcoBOP3612	current		(pymea-
property), 407	current	_uc_1 angc sure.instruments.agilent.Agilent34450A	* *
current (pymeasure.instruments.keysight.keysightE36312.	A Voltage(	~ ~	1
property), 420	_	* * * ·	(рутеа-
current (pymeasure.instruments.keysight.keysightE3631A			
property), 428	. vonage er	erty), 324	prop
current (pymeasure.instruments.keysight.KeysightN5767A	Current		(pymea-
property), 411	1 Current	_uc_range sure.instruments.keithley.KeithleyDMM	
current (pymeasure.instruments.novanta.Fpu60 prop-		property), 385	10300
erty), 492			(pymea-
current (pymeasure.instruments.oxfordinstruments.mercu			
property), 503	ryiic.mcan	erty), 324	ргор
current (pymeasure.instruments.rohdeschwarz.hmp.HMP	4040rent	• * * *	(pymea-
property), 543	/Cubi circ.	sure.instruments.keithley.KeithleyDMM	
current (pymeasure.instruments.siglenttechnologies.sigler	nt sndhasa		10300
property), 546			(рутеа-
current (pymeasure.instruments.srs.ldc500series.LDC500			4.5
property), 568	DETIESED	property), 386	10300
current (pymeasure.instruments.srs.ldc500series.LDC500	) Savina (PA)	* * * ·	(pymea-
property), 570	Jeura Merre	_ac_resoruction sure.instruments.agilent.Agilent34450A	
current (pymeasure.instruments.srs.ldc500series.LDC500	)SoriosTF(		1
			(mym a a
property), 571		<u> </u>	(pymea-
current (pymeasure.instruments.tdk.tdk_gen40_38.TDK_	Gen40_38		1
property), 590	C00-65	property), 163	(
current (pymeasure.instruments.tdk.tdk_gen80_65.TDK_	Gendbear.		(pymea-
property), 595		sure.instruments.teledyne.teledyne_osc	uioscope.1eieayneOsciiios
current (pymeasure.instruments.texio.TexioPSW360L30		property), 607	,
property), 623			(pymea-
current (pymeasure.instruments.toptica.ibeamsmart.IBea		sure.instruments.teledyne.teledyneMAU	)I. IeleayneMAUICnannel
property), 634		property), 612	(
current_ac (pymeasure.instruments.agilent.Agilent34410			(pymea-
property), 161		sure.instruments.temptronic.ATSBase	prop-
current_ac (pymeasure.instruments.agilent.Agilent34450		erty), 614	
property), 163	current	_dc (pymeasure.instruments.agilent.Agi	tient34410A
current_ac (pymeasure.instruments.hp.HP34401A		property), 161	D244014
property), 264	current	* * *	P34401A
current_ac_auto_range (pymea-		property), 264	
sure.instruments.agilent.Agilent34450A	current	_	(рутеа-
property), 163		sure.instruments.keithley.Keithley2000	prop-
current_ac_bandwidth (pymea-		erty), 324	,
sure.instruments.keithley.Keithley2000 prop-	current	_	(pymea-
erty), 324		sure.instruments.keithley.KeithleyDMM	16500
current_ac_bandwidth (pymea-		property), 386	,
sure.instruments.keithley.KeithleyDMM6500	current		(pymea-
property), 385		sure.instruments.keithley.Keithley2450	prop-
current_ac_digits (pymea-		erty), 353	
sure.instruments.keithley.Keithley2000 prop-	current		(рутеа-
erty) 324		sure instruments keithley Keithley 2450	pron-

	erty), 354			erty), 325	
	• **	(рутеа-	current	27.	(рутеа-
current.	sure.instruments.keithley.Keithley2450		carrent	_nprc sure.instruments.keithley.Keithley2400	4.2
	erty), 354	ргор		erty), 345	ргор
current	• **	(рутеа-	current	- ·	(рутеа-
	_ ·			_nprc sure.instruments.keithley.Keithley2450	4 -
		024A.SM U	Channei		prop-
	method), 134	(		erty), 354	(
	_fixed_pulsed_sweep()				(pymea-
		024X.SMU	Cnannei	sure.instruments.keithley.Keithley6517	В
	method), 134	,		property), 379	,
current.		(рутеа-	current	<del>-</del>	(pymea-
	sure.instruments.aimtti.aimttiPL.PLCh	annel		sure.instruments.keithley.KeithleyDAQ	6510
	property), 216			property), 401	
current.	_limit	(pymea-	current	_nplc	(pymea-
	sure.instruments.eurotest.EurotestHPP	120256		sure.instruments.keithley.KeithleyDMM	16500
	property), 255			property), 386	
current.	_limit	(рутеа-	current	_output_off_state	(pymea-
	sure. instruments. keithley. keithley 2200.	<i>PSChanne</i>	l	sure.instruments.keithley.Keithley2450	prop-
	property), 334			erty), 354	
current.		(pymea-	current	_ppm(pymeasure.instruments.danfysik.l	Danfysik8500
	sure.instruments.keithley.Keithley2260			property), 249	
	property), 336				(pymea-
current.		(рутеа-		sure.instruments.advantest.advantestR0	A .
current.	sure.instruments.keysight.keysightE363	4.2			2 111.51.10 01.01.01.0
	property), 420	11211. VOIIU	_		(рутеа-
current.		(рутеа-		_puiseu_sweep() sure.instruments.advantest.advantestR0	4.7
Current.		* *			124A.SM O Channel
		1A. voitag			(
	property), 429	,	current	• **	(pymea-
current.		(pymea-	CDDC	sure.instruments.advantest.advantestR6	024X.SMUCnanne
	sure.instruments.siglenttechnologies.sig	glent_spab			,
	property), 546	,		• **	(рутеа-
current.		(рутеа-		sure.instruments.advantest.advantestResize the state of	524X.SMUChanne
	sure.instruments.srs.ldc500series.LDC	500Series1	LD	method), 135	
	property), 568		current	_	(pymea-
current.		(рутеа-		sure.instruments.agilent.Agilent34450A	1
	sure.instruments.srs.ldc500series.LDC	500Series1	PD	property), 163	
	property), 570		current	_range	(pymea-
current.	_limit	(рутеа-		sure.instruments.agilent.AgilentB2981	prop-
	sure.instruments.srs.ldc500series.LDC	500Series'i	TEC	erty), 204	
	property), 571		current	_range	(pymea-
current		(pymea-		sure.instruments.aimtti.aimttiPL.PLCh	annel
	 sure.instruments.texio.TexioPSW360L3	* *		property), 216	
	property), 623		current		(pymea-
current.		(рутеа-	00111	sure.instruments.eurotest.EurotestHPP	4 *
current.	sure.instruments.yokogawa.YokogawaC	* ·		property), 255	120230
	property), 639	35200	current		(pymea-
	_measured	(рутеа-	Current	5	* *
Current.		4.2		sure.instruments.keithley.Keithley2000	prop-
	sure.instruments.oxfordinstruments.IPS	0120_10	G117070 '	erty), 325	(m) m a c
	property), 499	,	current	_	(рутеа-
current.		(рутеа-		sure.instruments.keithley.Keithley2400	prop-
	sure.instruments.agilent.agilent4156.Si	MU		erty), 345	(
	property), 169		current	_	(рутеа-
current.	_	(рутеа-		sure.instruments.keithley.Keithley2450	prop-
	sure instruments keithley Keithley 2000	nron-		erty) 354	

current_range		current	_setpoint	(pymea-
sure.instruments.keithley.Keithley6517	7B		sure.instruments.srs.ldc500series.LDC	C500SeriesPD
property), 379	,		property), 570	/
current_range		current	_setpoint	(pymea-
sure.instruments.keithley.KeithleyDAQ	96510		sure.instruments.srs.ldc500series.LDC	300SeriesTEC
property), 401			property), 571	
current_range		current	_setpoint	(рутеа-
sure.instruments.keithley.KeithleyDMic	M6500		sure.instruments.tdk.tdk_gen40_38.TD	0K_Gen40_38
property), 386			property), 590	
current_range	(рутеа-	current	_setpoint	(рутеа-
sure.instruments.keysight.KeysightN57	767A		sure.instruments.tdk.tdk_gen80_65.TD	K_Gen80_65
property), 411			property), 595	
current_range	(рутеа-	current	_source()	(рутеа-
sure.instruments.srs.ldc500series.LDC	C500SeriesI	LD	sure.instruments.advantest.advantestR	624X.SMUChannel
property), 569			method), 133	
current_reference	(рутеа-	current	_step	(рутеа-
sure.instruments.keithley.Keithley2000	prop-		sure.instruments.rohdeschwarz.hmp.H	MP4040
erty), 325			property), 543	
current_relative	(рутеа-		_sweep()	(pymea-
sure.instruments.keithley.KeithleyDM			sure.instruments.advantest.advantestRe	
property), 386			method), 134	
current_relative_enabled	(pymea-		_to_max()	(рутеа-
sure.instruments.keithley.KeithleyDM		0012 2 0110	sure.instruments.rohdeschwarz.hmp.H	
property), 386	110200		method), 543	
current_resolution	(mymea-		_to_min()	(рутеа-
sure.instruments.agilent.Agilent34450		Current	_to_min() sure.instruments.rohdeschwarz.hmp.H	
property), 163	71		method), 543	W11 4040
<pre>current_set_random_memory()</pre>	(рутеа-	Current	Range (class in	рутеа-
sure.instruments.advantest.advantestR	624X.SMU	Channel	sure.instruments.advantest.advantestR	624X),
method), 136				
			139	
current_setpoint	(рутеа-	curve_b	139 uffer_bits	(рутеа-
			uffer_bits	4 *
sure. instruments. a gilent. a gilent E 5270		ınnel	uffer_bits sure.instruments.signalrecovery.DSP7.	4 *
sure.instruments.agilent.agilentE5270 property), 214	B.SMUCha	ınnel	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553	225
<pre>sure.instruments.agilent.agilentE5270 property), 214 current_setpoint</pre>	B.SMUCho (pymea-	ınnel	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553 uffer_bits	(pymea-
sure.instruments.agilent.agilentE5270 property), 214 current_setpoint sure.instruments.danfysik.Danfysik850	B.SMUCho (pymea-	<i>nnel</i> curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7.	(pymea-
sure.instruments.agilent.agilentE5270 property), 214 current_setpoint sure.instruments.danfysik.Danfysik850 property), 249	B.SMUCho (pymea- 90	<i>nnel</i> curve_b	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553 uffer_bits sure.instruments.signalrecovery.DSP7 property), 559	225 (pymea- 265
sure.instruments.agilent.agilentE5270 property), 214 current_setpoint sure.instruments.danfysik.Danfysik850 property), 249 current_setpoint	B.SMUCho (pymea- )0 (pymea-	<i>nnel</i> curve_b	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553 uffer_bits sure.instruments.signalrecovery.DSP7 property), 559 uffer_interval	225 (pymea- 265 (pymea-
sure.instruments.agilent.agilentE5270 property), 214 current_setpoint sure.instruments.danfysik.Danfysik850 property), 249 current_setpoint sure.instruments.keithley.keithley4200	B.SMUCho (pymea- )0 (pymea-	<i>nnel</i> curve_b	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553 uffer_bits sure.instruments.signalrecovery.DSP7 property), 559 uffer_interval sure.instruments.signalrecovery.DSP7	225 (pymea- 265 (pymea-
sure.instruments.agilent.agilentE5270 property), 214 current_setpoint sure.instruments.danfysik.Danfysik850 property), 249 current_setpoint sure.instruments.keithley.keithley4200 property), 405	B.SMUCho (pymea- 00 (pymea- .SMU	<i>unnel</i> curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553	225 (pymea- 265 (pymea- 225
sure.instruments.agilent.agilentE5270 property), 214 current_setpoint sure.instruments.danfysik.Danfysik850 property), 249 current_setpoint sure.instruments.keithley.keithley4200 property), 405 current_setpoint	B.SMUCho (pymea- 00 (pymea- .SMU (pymea-	<i>unnel</i> curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval	225 (pymea- 265 (pymea- 225 (pymea-
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361	B.SMUCho (pymea- 00 (pymea- .SMU (pymea-	<i>unnel</i> curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7.	225 (pymea- 265 (pymea- 225 (pymea-
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407	B.SMUCho (pymea- )0 (pymea- !SMU (pymea- !2	unnel  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553 uffer_bits sure.instruments.signalrecovery.DSP7 property), 559 uffer_interval sure.instruments.signalrecovery.DSP7 property), 553 uffer_interval sure.instruments.signalrecovery.DSP7 property), 559	(pymea- 265 (pymea- 225 (pymea- 265
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint	B.SMUCho (pymea- )0 (pymea- ).SMU (pymea- )2 (pymea-	unnel  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7 property), 553 uffer_bits sure.instruments.signalrecovery.DSP7 property), 559 uffer_interval sure.instruments.signalrecovery.DSP7 property), 553 uffer_interval sure.instruments.signalrecovery.DSP7 property), 553 uffer_interval sure.instruments.signalrecovery.DSP7 property), 559 uffer_length	(pymea- 265 (pymea- 225 (pymea- 265 (pymea-
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP	B.SMUCho (pymea- )0 (pymea- ).SMU (pymea- )2 (pymea-	unnel  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7.	(pymea- 265 (pymea- 225 (pymea- 265 (pymea-
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499	B.SMUCho (pymea- 200 (pymea- 2120 (pymea- 22 (pymea- 2120_10	unnel  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553	(pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint	B.SMUCho (pymea- 200) (pymea- 20) (pymea- 21) (pymea- 21) (pymea-	unnel  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length	225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 225
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman	B.SMUCho (pymea- 200) (pymea- 20) (pymea- 21) (pymea- 21) (pymea-	unnel  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7.	225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 225
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman property), 565	B.SMUCho (pymea- )0 (pymea- ).SMU (pymea- 2 (pymea- S120_10 (pymea- ;XRV	unnel  curve_b  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 559	(pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 225
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman property), 565  current_setpoint	B.SMUCho (pymea- )00 (pymea- ).SMU (pymea- 22 (pymea- S120_10 (pymea- XRV (pymea-	unnel  curve_b  curve_b  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_status	(pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 265
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman property), 565  current_setpoint sure.instruments.spellmanhv.spellman	B.SMUCho (pymea- )00 (pymea- ).SMU (pymea- 22 (pymea- S120_10 (pymea- XRV (pymea-	unnel  curve_b  curve_b  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_status sure.instruments.signalrecovery.DSP7.	(pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 265
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman property), 565  current_setpoint sure.instruments.spellmanhv.spellman property), 567	B.SMUCho (pymea- )0 (pymea- ).SMU (pymea- )2 (pymea- S120_10 (pymea- XRV (pymea- XRV.Unsca	curve_b  curve_b  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_status sure.instruments.signalrecovery.DSP7. property), 559 uffer_status sure.instruments.signalrecovery.DSP7. property), 553	(pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 265
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman property), 565  current_setpoint sure.instruments.spellmanhv.spellman property), 567  current_setpoint	B.SMUCho (pymea- )0 (pymea- ).SMU (pymea- )2 (pymea- S120_10 (pymea- XRV (pymea- XRV (pymea- XRV.Unsca	curve_b  curve_b  curve_b  curve_b  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_status sure.instruments.signalrecovery.DSP7. property), 553 uffer_status	(pymea- 225 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225
sure.instruments.agilent.agilentE5270 property), 214  current_setpoint sure.instruments.danfysik.Danfysik850 property), 249  current_setpoint sure.instruments.keithley.keithley4200 property), 405  current_setpoint sure.instruments.kepco.KepcoBOP361 property), 407  current_setpoint sure.instruments.oxfordinstruments.IP property), 499  current_setpoint sure.instruments.spellmanhv.Spellman property), 565  current_setpoint sure.instruments.spellmanhv.spellman property), 567	B.SMUCho (pymea- )0 (pymea- ).SMU (pymea- )2 (pymea- S120_10 (pymea- XRV (pymea- XRV (pymea- XRV.Unsca	curve_b  curve_b  curve_b  curve_b  curve_b  curve_b  curve_b	uffer_bits sure.instruments.signalrecovery.DSP7. property), 553 uffer_bits sure.instruments.signalrecovery.DSP7. property), 559 uffer_interval sure.instruments.signalrecovery.DSP7. property), 553 uffer_interval sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 553 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_length sure.instruments.signalrecovery.DSP7. property), 559 uffer_status sure.instruments.signalrecovery.DSP7. property), 559 uffer_status sure.instruments.signalrecovery.DSP7. property), 553	(pymea- 225 (pymea- 225 (pymea- 265 (pymea- 225 (pymea- 265 (pymea- 225

CustomIntEnum (class in pymea- sure.instruments.agilent.agilentB1500), 194	<pre>dap_unit (pymeasure.instruments.ptw.ptwDIAMENTOR</pre>
cw_frequency (pymea-	data (pymeasure.experiment.experiment.Experiment
sure.instruments.rohdeschwarz.sfm.SFM	property), 64 data (pymeasure.instruments.agilent.Agilent8722ES
property), 529 cw_mode_enabled (pymea-	property), 152
	property), 132 r <b>chwe a</b> t <b>(p)ymens</b> lure.instruments.agilent.agilentE5062A.VNAChannel
property), 243	property), 158
* * *	data() (pymeasure.display.widgets.sequencer_widget.SequencerTreeMode
sure.instruments.anritsu.anritsuMS464xB.Measu	
property), 243	data() (pymeasure.display.widgets.table_widget.PandasModelBase
CXN (class in pymeasure.instruments.tcpowerconversion),	method), 89
585	data_arb() (pymeasure.instruments.agilent.Agilent33500
CXN.Status (class in pymea-	method), 176
sure.instruments.tcpowerconversion), 586	data_arb() (pymeasure.instruments.agilent.agilent33500.Agilent33500Ch
cycling_enable (pymea-	method), 179
sure.instruments.temptronic.ATSBase prop-	data_arb() (pymeasure.instruments.keysight.Keysight81160A
erty), 615	method), 435
cycling_stopped() (pymea-	data_buffer_size (pymea-
sure.instruments.temptronic.ATSBase method),	sure.instruments.agilent.AgilentB2981 prop-
615	erty), 204
	data_complex (pymea-
D	sure.instruments.agilent.Agilent8722ES
dac1 (pymeasure.instruments.ametek.Ametek7270 prop-	property), 152
erty), 220	data_complex (pymeasure.instruments.hp.HP8753E
dac1 (pymeasure.instruments.signalrecovery.DSP7225	property), 307
property), 553	data_drawing_enabled (pymea-
dac1 (pymeasure.instruments.signalrecovery.DSP7265	sure.instruments.anritsu.AnritsuMS464xB
property), 560	property), 238
dac1 (pymeasure.instruments.srs.SR830 property), 576	data_format(pymeasure.instruments.keithley.KeithleyDMM6500
dac1 (pymeasure.instruments.srs.SR860 property), 580	property), 386
dac2 (pymeasure.instruments.ametek.Ametek7270 prop-	data_format() (pymea-
erty), 221	sure.instruments.agilent.agilentB1500.AgilentB1500
dac2 (pymeasure.instruments.signalrecovery.DSP7225	method), 186
property), 553	data_log_magnitude (pymea- sure.instruments.agilent.Agilent8722ES
dac2 (pymeasure.instruments.signalrecovery.DSP7265	property), 152
property), 560 dac2 (pymeasure.instruments.srs.SR830 property), 576	
	sure.instruments.agilent.Agilent8722ES
dac2 (pymeasure.instruments.srs.SR860 property), 580 dac3 (pymeasure.instruments.ametek.Ametek7270 prop-	property), 152
erty), 221	data_memory_a_condition (pymea-
dac3 (pymeasure.instruments.signalrecovery.DSP7265	sure.instruments.anritsu.AnritsuMS9710C
property), 560	property), 231
dac3 (pymeasure.instruments.srs.SR830 property), 576	data_memory_a_size (pymea-
dac3 (pymeasure.instruments.srs.SR860 property), 580	sure.instruments.anritsu.AnritsuMS9710C
dac4 (pymeasure.instruments.ametek.Ametek7270 prop-	property), 231
erty), 221	data_memory_a_values (pymea-
dac4 (pymeasure.instruments.signalrecovery.DSP7265	sure.instruments.anritsu.AnritsuMS9710C
property), 560	property), 231
dac4 (pymeasure.instruments.srs.SR830 property), 576	data_memory_b_condition (pymea-
dac4 (pymeasure.instruments.srs.SR860 property), 580	sure.instruments.anritsu.AnritsuMS9710C
Danfysik8500 (class in pymea-	property), 231
sure.instruments.danfysik), 249	data_memory_b_size (pymea-
••	sure.instruments.anritsu.AnritsuMS9710C

property), 231		method), 454	
data_memory_b_values (pymea-	dc_volta	${\sf lge}(pymeasure. instruments. tcpowerous)$	conversion.CXN
sure.instruments.anritsu.AnritsuMS9710C	_	property), 586	
property), 231	dcmode	(pymeasure.instruments.srs.SR860	property),
data_memory_select (pymea-	•	580	
sure.instruments.anritsu.AnritsuMS9710C	DCXS (clas	ss in pymeasure.instruments.aja), 21	8
property), 231	deactiva	ite_all()	(pymea-
data_memory_select (pymea-		sure.instruments.hp.HP11713A	method),
sure.instruments.anritsu.AnritsuMS9740A		310	
property), 233	deactiva	ite_marker()	(pymea-
data_phase (pymeasure.instruments.agilent.Agilent8722	EES .	sure.instruments.hp.hp856Xx.HP856	6Xx
property), 152		method), 290	
	decimati	on (pymeasure.instruments.redpitay	a.redpitaya_scpi.RedPitayaS
sure.instruments.agilent.agilent4156.Agilent415		property), 524	
property), 167	decode()	* * * '	.Racal1992
data_volatile_clear() (pymea-		static method), 519	
sure.instruments.agilent.Agilent33500 method),			(рутеа-
176		sure.instruments.keysight.KeysightD	
data_volatile_clear() (pymea-		method), 409	50/11/020
sure.instruments.agilent.agilent33500.Agilent33			(рутеа-
method), 179		sure.instruments.lecroy.LeCroyT3DS	
data_volatile_clear() (pymea-		method), 462	301204
	default_		(mymag
sure.instruments.keysight.Keysight81160A		= ''	(pymea-
method), 435		sure.instruments.teledyne.TeledyneO	озстоѕсоре
datablock_header_format (pymea-		method), 603	(
sure.instruments.anritsu.AnritsuMS464xB		arbitary_waveform()	(pymea-
property), 238		sure.instruments.keithley.Keithley62	21
datablock_numeric_format (pymea-		method), 372	,
sure.instruments.anritsu.AnritsuMS464xB		oosition()	(pymea-
property), 239		sure.instruments.newport.esp300.Ax	is method),
datafile_frequency_unit (pymea-		477	
sure.instruments.anritsu.AnritsuMS464xB	_	ymeasure.instruments.hp.HP3437A	property),
property), 239		267	
datafile_include_heading (pymea-	delay (p	ymeasure.instruments.hp.HP6632A	property),
sure.instruments.anritsu.AnritsuMS464xB		318	
property), 239	delay_en	${\tt lable}\ (pymeasure. instruments. racal.$	.Racal1992
datafile_parameter_format (pymea-		property), 519	
sure.instruments.anritsu.AnritsuMS464xB	delay_ti	me (pymeasure.instruments.agilent.a	agilent4156.Agilent4156
property), 239	i	property), 167	
$\verb"date" (pymeasure.instruments.redpitaya.redpitaya\_scpi.Redpitay$	ed <b>Ried</b> yayScypii	ime (pymeasure.instruments.racal.	.Racal1992
property), 524	i	property), 519	
date (pymeasure.instruments.rohdeschwarz.sfm.SFM	delete_c	channel()	(pymea-
property), 529		sure.instruments.rohdeschwarz.fsser	ries.FSW
DBM (pymeasure.instruments.hp.hp856Xx.AmplitudeUnits	' i	method), 542	
attribute), 300		lata_file()	(рутеа-
DBMV (pymeasure.instruments.hp.hp856Xx.AmplitudeUnit.	's	sure.instruments.anritsu.AnritsuMS4	464xB
attribute), 300		method), 239	
DBUV (pymeasure.instruments.hp.hp856Xx.AmplitudeUnit.			(рутеа-
attribute), 300		sure.instruments.anritsu.AnritsuMS4	
DC (pymeasure.instruments.hp.hp856Xx.CouplingMode		method), 239	
attribute), 301	delete_t		(рутеа-
dc (pymeasure.instruments.rigol.rigol_dg800.VoltageCha		sure.instruments.yokogawa.aq6370s	
property), 524		method), 640	
dc_mode() (pymeasure.instruments.lakeshore.LakeShore			(рутеа-
	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~		

sure.instruments.keysight.keysight8116	0A.Keysig	ht81160AChilmtel), 281	
method), 440		demodulation_time	(pymea-
<pre>delta_abort()</pre>	(рутеа-	sure.instruments.hp.hp856Xx.HP856X	x at-
sure.instruments.keithley.Keithley6221	**	tribute), 281	
method), 372		DemodulationMode (class in	рутеа-
delta_arm() (pymeasure.instruments.keithley.K	eithley622		F.)
method), 372	ienney 022	deposition_time_min	(рутеа-
delta_buffer_points	(рутеа-	sure.instruments.aja.DCXS property),	
sure.instruments.keithley.Keithley6221		deposition_time_sec	(pymea-
erty), 372	ргор-	sure.instruments.aja.DCXS property),	4.5
* *	(1221111 0 0		
	(рутеа-	derivative_action_time	(pymea-
sure.instruments.keithley.Keithley6221	prop-	sure.instruments.oxfordinstruments.IT	2303
erty), 372	,	property), 495	ano co
	(рутеа-	detectedfrequency (pymeasure.instruments.s	rs.SR860
sure.instruments.keithley.Keithley6221	prop-	property), 580	
erty), 373		DetectionModes (class in	рутеа-
delta_connected	(pymea-	sure.instruments.hp.hp856Xx), 301	
sure.instruments.keithley.Keithley6221	prop-	detector_bandwidth	(pymea-
erty), 373		sure.instruments.hp.HP34401A p	roperty),
delta_cycles	(рутеа-	264	
sure.instruments.keithley.Keithley6221		detector_bandwidth	(pymea-
erty), 373	1 1	sure.instruments.keithley.KeithleyDMM	
delta_delay(pymeasure.instruments.keithley.K	eithlev622		10000
property), 373	ienney 022	detector_mode	(рутеа-
delta_high_source	(рутеа-	sure.instruments.hp.hp856Xx.HP856X	4.7
sure.instruments.keithley.Keithley6221		tribute), 275	<i>x</i>
	ргор-		(mayor a a
erty), 373	(	determine_valid_channels()	(pymea-
delta_low_source	(рутеа-	sure.instruments.keithley.Keithley2700	
sure.instruments.keithley.Keithley6221	prop-	method), 366	c
erty), 373		deviation(pymeasure.instruments.rohdeschwa	rz.sfm.Sound_Channel
delta_measurement_sets	(рутеа-	property), 537	
sure.instruments.keithley.Keithley6221	prop-		(pymea-
erty), 373		$sure.instruments.thyracont.smartline\_v$	v2.SmartlineV2
delta_sense(pymeasure.instruments.keithley.K	Keithley622	21 property), 630	
property), 373		device_operation_enable_register	(pymea-
<pre>delta_start()</pre>	(pymea-	sure.instruments.advantest.advantestRest	624X.AdvantestR624X
sure.instruments.keithley.Keithley6221		property), 125	
method), 373		device_operation_register	(pymea-
delta_unit(pymeasure.instruments.keithley.Ke	eithlev6221	l sure.instruments.advantest.advantestR	624X.AdvantestR624X
property), 373		property), 126	
delta_values	(рутеа-	device_serial	(pymea-
sure.instruments.keithley.Keithley6221	4 2	sure.instruments.thyracont.smartline_\tau	
	ргор-		v2.Smarttinev2
erty), 373	(	property), 630	Jan 74h MVC074D
demand_current	(pymea-	device_type (pymeasure.instruments.mksinst.m	IKS9/40.MIKS9/4D
sure.instruments.oxfordinstruments.IPS	\$120_10	property), 475	11002
property), 499		device_type (pymeasure.instruments.racal.Ra	acal1992
demand_field	(рутеа-	property), 519	
sure.instruments.ox for dinstruments.IPS	\$120_10	device_type (pymeasure.instruments.thyracon	t.smartline_v1.SmartlineV
property), 499		property), 628	
	(pymea-	<pre>device_type (pymeasure.instruments.thyracon)</pre>	t.smartline_v2.Smartline\
sure.instruments.hp.hp856Xx.HP856X	x at-	property), 630	
tribute), 281		device_version	(pymea-
demodulation_mode	(рутеа-	sure.instruments.thyracont.smartline_v	v2.SmartlineV2
sure instruments hn hn856Xx HP856X		property) 630	

DG800 (class in pymeasure.instruments.rigol), 524	method), 505
	e() (pymeasure.instruments.toptica.ibeamsmart.IBeamSmart
sure.instruments.advantest.advantestR624X.AdvantestR624	
property), 125 disable	
digital_reset() (pymea-	sure.instruments.agilent.agilent4156.Agilent4156
sure.instruments.redpitaya.redpitaya_scpi.RedPitayaScpi	
	e_amplitude_modulation() (pymea-
digitize() (pymeasure.instruments.keysight.KeysightDSOX1102G	ŭ ŭ
method), 409	method), 150
digits(pymeasure.instruments.keithley.KeithleyDMM6500disable	
property), 386	sure.instruments.agilent.Agilent8722ES
diode (pymeasure.instruments.agilent.Agilent34450A	method), 152
	e_bias() (pymeasure.instruments.srs.SR570
diode (pymeasure.instruments.keithley.KeithleyDMM6500	method), 573
property), 386 disable diode_bias (pymeasure.instruments.keithley.KeithleyDMM6500	e_buffer() (pymea- sure.instruments.keithley.Keithley2000
property), 386	method), 325
diode_nplc(pymeasure.instruments.keithley.KeithleyDMM665646ble	
property), 386	e_buffer() (pymea- sure.instruments.keithley.Keithley2182
dioN (pymeasure.instruments.redpitaya.redpitaya_scpi.RedPitayaSc	
	e_buffer() (pymea-
dioP (pymeasure.instruments.redpitaya.redpitaya_scpi.RedPitayaSc	A.
attribute), 523	method), 345
dip_search(pymeasure.instruments.anritsu.AnritsuMS971 <b>01</b> Csable	
property), 231	sure.instruments.keithley.Keithley2450
direction(pymeasure.instruments.anaheimautomation.DPSeriesM	· · · · · · · · · · · · · · · · · · ·
* *	e_buffer() (pymea-
direction(pymeasure.instruments.mksinst.mksinst.RelayChannel	
property), 471	method), 366
directory (pymeasure.display.widgets.fileinput_widget.FileInpubWe	dmiffer() (pymea-
property), 82	sure.instruments.keithley.Keithley6221
DirectoryLineEdit (class in pymea-	method), 373
sure.display.widgets.directory_widget), 82 disable	e_buffer() (pymea-
${\tt disable} \ ({\it pymeasure.instruments.agilent.agilent4156.Agilent Measure.instruments.agilent.agilent4156.Agilent Measure.instruments.agilent.agilent4156.Agilent Measure.instruments.agilent.agilent4156.Agilent Measure.instruments.agilent.agilent.agilent4156.Agilent Measure.instruments.agilent.agil$	re <b>smarattaktrnumee</b> nts.keithley.Keithley6517B
property), 169	method), 379
disable() (pymeasure.instruments.agilent.Agilent8257D disable	
method), 150	sure.instruments.keithley.KeithleyDAQ6510
disable() (pymeasure.instruments.agilent.agilentB1500.SMU	
	e_control() (pymea-
disable() (pymeasure.instruments.anritsu.AnritsuMG3692C	sure.instruments.oxfordinstruments.IPS120_10
method), 230	method), 499
disable() (pymeasure.instruments.danfysik.Danfysik8500 disable	
method), 250	sure.instruments.novanta.Fpu60 method),
disable() (pymeasure.instruments.deltaelektronika.SM7045D	492
	e_filter() (pymea-
disable() (pymeasure.instruments.keithley.keithley4200.SMU method), 406	sure.instruments.keithley.Keithley2000 method), 325
disable() (pymeasure.instruments.keysight.KeysightN57674 sable	
method), 411	sure.instruments.keithley.KeithleyDMM6500
disable() (pymeasure.instruments.newport.ESP300	method), 386
* *	e_filter() (pymea-
disable() (pymeasure.instruments.newport.esp300.Axis	sure.instruments.keithley.keithleyDMM6500.ScannerCard2000Cl
method), 477	method), 393
disable() (pymeasure.instruments.parker.ParkerGV6 disable	

	sure.instruments.agilent.Agilent8257D		property), 435	
	method), 150	displa	ay (pymeasure.instruments.tdk.tdk_gen-	40_38.TDK_Gen40_38
disable.	_modulation() (pyme	a-	property), 590	
	sure.instruments.agilent.Agilent8257D	displa	ay (pymeasure.instruments.tdk.tdk_gen	80_65.TDK_Gen80_65
	<i>method</i> ), 150		property), 595	
disable.	_offset_current() (pyme	a- displa	ay (pymeasure.instruments.teledyne.tele	$edyne\_oscilloscope. Teledyne Oscilloscope$
	sure.instruments.srs.SR570 method), 573		property), 608	
disable	_output_trigger() (pyme	a- displa	ay_active (pymeasure.instruments.hp	p.HP6632A
	sure.instruments.keithley.Keithley2400		property), 318	
	method), 345		ay_all_segments_enabled	(рутеа-
disable	_output_trigger() (pyme		sure.instruments.hp.HP437B proper	
	sure.instruments.keithley.Keithley6221	displa	ay_brightness	(рутеа-
	method), 373		sure.instruments.keithley.Keithley23	06 prop-
disable.	_persistent_mode() (pyme		erty), 342	
	$sure. instruments. ox for dinstruments. IPS 120\_1$	$\theta$ displa		(рутеа-
	method), 500		sure.instruments.keithley.Keithley23	06 prop-
disable.	_persistent_switch() (pyme		erty), 342	
	sure.instruments.ami.AMI430 method), 223		ay_closed_channels()	(pymea-
	_pulse_modulation() (pyme	a-	sure.instruments.keithley.Keithley27	700
	sure.instruments.agilent.Agilent8257D		method), 366	
	method), 150	_	ay_data	(pymea-
	_reference() (pyme	a-	sure.instruments.thyracont.smartline	e_v2.SmartlineV2
	sure.instruments.keithley.Keithley2000		property), 630	
	method), 325	_	ay_enabled	(pymea-
disable.	_relative() (pyme	a-	sure.instruments.advantest.advantes	tR624X.AdvantestR624X
	sure.instruments.keithley.KeithleyDMM6500	12 1	property), 126	
	method), 387	_	ay_enabled	(pymea-
disable.			sure.instruments.hp.HP34401A	property),
	sure.instruments.anapico.APSIN12G method		264	l IID427D
dicable	_source() (pyme		ay_enabled (pymeasure.instruments.i	пр.пР43/Б
uisabie.			<pre>property), 311 ay_enabled</pre>	(mayor a a
	sure.instruments.keithley.Keithley2400 method), 346	urspr	sure.instruments.keithley.Keithley21	(pymea-
		a	erty), 396	62 <i>prop-</i>
	_source() (pymed sure.instruments.keithley.Keithley2450		ay_enabled	(рутеа-
	method), 354	urspr	sure.instruments.keithley.Keithley22	4.5
	_source() (pymed	a_	erty), 333	оо ргор
	sure.instruments.keithley.Keithley2510		ay_enabled	(рутеа-
	method), 360	атэрт	sure.instruments.keithley.Keithley23	
	_source() (pymed	a-	erty), 342	oo p.op
urbubre.	sure.instruments.keithley.Keithley6221		ay_enabled	(рутеа-
	method), 373	агорг	sure.instruments.keithley.Keithley24	
	_source() (pymed	a-	erty), 346	oo p.op
	sure.instruments.keithley.Keithley6517B		ay_enabled	(рутеа-
	method), 379		sure.instruments.keithley.Keithley62	
	_source() (pyme	a-	erty), 373	r · r
	sure.instruments.yokogawa.Yokogawa7651		ay_estimates()	(рутеа-
	method), 638	-1-	sure.display.widgets.estimator_widg	
	_temperature_protection() (pyme	a-	method), 82	O ·
	sure.instruments.keithley.Keithley2510		ay_filter_enabled	(рутеа-
	method), 360	•	sure.instruments.lakeshore.LakeSho	• •
	(pymeasure.instruments.agilent.Agilent3350	00	property), 451	
	property), 176		ay_layout	(рутеа-
display	(pymeasure.instruments.keysight.Keysight811)	60A	sure.instruments.agilent.AgilentE50	62A

property), 156			269			
display_layout	(рутеа-	display				(рутеа-
sure.instruments.agilent.agilentE50622	4 .			s.thyracor	ıt.smartline	_v1.SmartlineV1
property), 158			property), 628	,		_
display_layout	(рутеа-					(рутеа-
sure.instruments.anritsu.AnritsuMS464				s.thyracor	ıt.smartline	_v2.SmartlineV2
property), 239			property), 630			
display_layout	(рутеа-					(рутеа-
sure.instruments.anritsu.anritsuMS464				s.lakeshor	re.LakeShor	
property), 243			property), 446			
display_line	(pymea-		_user_message	2		(рутеа-
sure.instruments.hp.hp856Xx.HP856X		1	sure.instrument		37B properi	* *
tribute), 278		display		T .	r	(pymea-
display_orientation	(рутеа-		sure.instrument	s.hp.HP34	4401A	
sure.instruments.thyracont.smartline_v		neV2	264			F F
property), 630			ed_text()			(рутеа-
display_output (pymeasure.instruments.hp.	HP437B		sure.instrument	s.keithlev.	KeithlevDM	* *
property), 311			method), 387	~		
display_parameter	(pymea-	do_fft(	) (pymeasure.ins	struments.	hp.hp856X2	x.HP856Xx
sure.instruments.teledyne.teledyne_osc					1 1	
property), 608	1	DockWid	_		in	рутеа-
display_parameter()	(рутеа-		sure.display.wid	dgets.dock	_widget), 8	
sure.instruments.lecroy.LeCroyT3DSO		DOR	(class		in	
method), 462			sure.instrument	s.advantes	st.advantest	R624X),
display_parameter()	(рутеа-		140			,,
sure.instruments.teledyne.TeledyneOsc		downloa	d_data()			(pymea-
method), 603	•		sure.instrument	s.keysight	.KeysightD	SOX1102G
display_parameters	(pymea-		method), 409			
sure.instruments.hp.hp856Xx.HP856X	x at-	downloa	d_image()			(рутеа-
tribute), 280			sure.instrument	s.keysight	.KeysightD.	SOX1102G
display_reset()	(pymea-		method), 409			
sure.instruments.hp.HP3478A method)	), 269	downloa	d_image()			(pymea-
display_screen	(pymea-		sure.instrument	s.lecroy.L	eCroyT3DS	O1204
sure.instruments.keithley.KeithleyDMM	M6500		method), 462			
property), 387		downloa	d_image()			(pymea-
display_text (pymeasure.instruments.hp.F	HP3478A		sure.instrument	s.teledyne	.TeledyneM	AUI
property), 269			method), 610			
		downloa	d_image()			(рутеа-
sure.instruments.keithley.Keithley2700	prop-		sure.instrument	s.teledyne	.TeledyneO	scilloscope
erty), 366			method), 603			
display_text_data	(рутеа-	downloa	d_waveform()			(рутеа-
sure.instruments.keithley.Keithley2200	prop-		sure.instrument	s.lecroy.L	eCroyT3DS	O1204
erty), 333			method), 462			
display_text_data	(рутеа-	downloa	d_waveform()			(pymea-
sure.instruments.keithley.Keithley2281	S		sure.instrument	s.teledyne	.TeledyneO	scilloscope
property), 339	,		method), 603			
display_text_data	(pymea-	DPSerie	sMotorControl	,	class in	рутеа-
sure.instruments.keithley.Keithley2306	prop-		sure.instrument			1), 224
erty), 342	,	DriverC		(class	in	рутеа-
display_text_enabled	(pymea-		sure.instrument	_		
sure.instruments.keithley.Keithley2306	prop-	dsp (p	ymeasure.instru	ments.spel	umanhv.Spe	eumanXRV
erty), 342	(	DCDZZZZ	property), 565	_		
display_text_no_symbol	4.0	DSP7225	(class		in	рутеа-
sure.instruments.hp.HP3478A p	roperty),		sure.instrument	s.signaire	covery), 33	1

DSP726	55 (class is sure.instruments.signalrecov	1 .	enable (pymeasure.instruments.edwards.Nxds property), 253
dut_co	nstant sure.instruments.temptronic.	(pymea-	
	erty), 615	• •	enable() (pymeasure.instruments.agilent.agilentB1500.SMU
dut_mo	de (pymeasure.instruments.to	emptronic.ATSBase	method), 190
	property), 615		enable() (pymeasure.instruments.anritsu.AnritsuMG3692C
dut_te	emperature	(pymea-	
	sure.instruments.temptronic. erty), 615	•	method), 250
	property), 615		e enable() (pymeasure.instruments.deltaelektronika.SM7045D method), 252
duty(p	ymeasure.instruments.tektroni. property), 599	x.afg3152c.AFG315	52@Ghhhhed) (pymeasure.instruments.keysight.KeysightN5767A method), 411
duty_c	cycle (pymeasure.instruments erty), 311	hp.HP437B prop-	enable() (pymeasure.instruments.newport.ESP300 method), 476
	erty), 272		enable() (pymeasure.instruments.newport.esp300.Axis method), 477
duty_c	= :	rigol.rigol_dg800.V	Vol <b>ægabLæg</b> nel (pymeasure.instruments.parker.ParkerGV6
1	property), 524	,	method), 505
duty_c	cycle_enabled	(pymea-	* *
duo 11	sure.instruments.hp.HP437Etime (pymeasure.instruments.		sure.instruments.siglenttechnologies.siglent_spdbase.SPDSingleC
uweii_	property), 150	uguem.Aguem0257	7D method), 546 enable_air_flow (pymea-
dvnami	.c_temperature_setpoint	(рутеа-	_ ·
	sure.instruments.temptronic.		
	erty), 615	1 1	enable_amplitude_modulation() (pymea-
_			sure.instruments.agilent.Agilent8257D
E			method), 150
echo (	(pymeasure.instruments.parker	:ParkerGV6 prop-	enable_averaging() (pymea-
	erty), 505		sure.instruments.agilent.Agilent8722ES
EC0560	(class in pymeasure.instrumer	its.temptronic), 622	method), 152
elapse	d_time	(pymea-	1 1 574
	sure.instruments.hp.hp856X	x.HP856Xx at-	
_	tribute), 280		enable_continous() (pymea- sure.instruments.toptica.ibeamsmart.IBeamSmart
electr	rical_units_enabled	(pymea-	1 1) 624
	sure.instruments.ptw.ptwUN 514	IDOS property),	enable_control() (pymea-
omorgo	ency_off()	(рутеа-	
emer ge	sure.instruments.eurotest.Eu		method), 500
	method), 255	101051111 1 120230	enable_filter() (pymea-
emissi	on (pymeasure.instruments.top	otica.ibeamsmart.IB	
	property), 634		method), 325
emissi	on_enabled	(рутеа-	
	sure.instruments.ipgphotonia erty), 322	cs.yar.YAR prop-	method), 387
emissi	on_enabled	(рутеа-	enable_filter() (pymea-
	sure.instruments.novanta.Fp 492	pu60 property),	method), 393
	(pymeasure.display.log.LogH		
emit()	(pymeasure.experiment.worke 71	rs.Worker method),	method), 545
emit_b	peep() (pymeasure.instrumethod), 308	ments.hp.HP8753E	g enable_low_freq_out() (pymea- sure.instruments.agilent.Agilent8257D

method), 150

enable\_offset\_current()

- · · · · · · · · · · · · · · · · · · ·	enablea (pymeasure.instruments.activetechnologies.AwG401x_AFG
sure.instruments.srs.SR570 method), 574	property), 112
enable_output() (pymea-	* · · · · · · · · · · · · · · · · · · ·
sure.instruments.siglenttechnologies.siglent_spdl	base.SPDCl <b>panpel</b> ty), 112
method), 547	enabled (pymeasure.instruments.agilent.agilent4284A.Agilent4284ASpot
<pre>enable_persistent_mode()</pre>	property), 200
sure.instruments.oxfordinstruments.IPS120_10	enabled (pymeasure.instruments.agilent.agilentE5270B.SMUChannel
method), 500	property), 214
enable_persistent_switch() (pymea-	enabled (pymeasure.instruments.aja.DCXS property),
sure.instruments.ami.AMI430 method), 223	219
· · · · · · · · · · · · · · · · · · ·	
-	enabled (pymeasure.instruments.hp.hp11713a.SwitchDriverChannel
sure.instruments.agilent.Agilent8257D	property), 310
method), 150	enabled (pymeasure.instruments.keithley.Keithley2260B
enable_pulsing() (pymea-	property), 336
	rt enabled (pymeasure.instruments.mksinst.mks937b.Relay
method), 634	property), 474
enable_reference() (pymea-	enabled (pymeasure.instruments.mksinst.mks974b.Relay
sure.instruments.keithley.Keithley2000	property), 476
method), 325	enabled (pymeasure.instruments.newport.esp300.Axis
enable_relative() (pymea-	property), 477
sure.instruments.keithley.KeithleyDMM6500	enabled (pymeasure.instruments.spellmanhv.spellmanXRV.Filament
method), 387	property), 567
	G enabled (pymeasure.instruments.srs.ldc500series.LDC500SeriesLD
method), 227	property), 569
enable_source() (pymea-	enabled (pymeasure.instruments.srs.ldc500series.LDC500SeriesTEC
sure.instruments.advantest.advantestR624X.Adva	
method), 119	enabled (pymeasure.instruments.texio.TexioPSW360L30
	= -
enable_source() (pymea-	property), 623
	Ucharbled (pymeasure.instruments.toptica.ibeamsmart.DriverChannel
method), 136	property), 635
enable_source() (pymea-	encoder_autocorrect (pymea-
sure.instruments.keithley.Keithley2400	sure. instruments. an a heimautomation. DPS eries Motor Controller
method), 346	property), 225
enable_source() (pymea-	encoder_delay (pymea-
sure.instruments.keithley.Keithley2450	sure. instruments. an aheim automation. DPS eries Motor Controller
method), 354	property), 225
enable_source() (pymea-	encoder_enabled (pymea-
sure.instruments.keithley.Keithley2510	sure. instruments. an aheim automation. DPS eries Motor Controller
method), 360	property), 225
	encoder_motor_ratio (pymea-
sure.instruments.keithley.Keithley6221	sure.instruments.anaheimautomation.DPSeriesMotorController
method), 373	property), 225
enable_source() (pymea-	encoder_retries (pymea-
sure.instruments.keithley.Keithley6517B	sure.instruments.anaheimautomation.DPSeriesMotorController
method), 380	property), 225
enable_source() (pymea-	* *
sure.instruments.yokogawa.Yokogawa7651	sure.instruments.anaheimautomation.DPSeriesMotorController
method), 638	property), 225
enable_temperature_protection() (pymea-	end_of_all_cycles() (pymea-
sure.instruments.keithley.Keithley2510	sure.instruments.temptronic.ATSBase method),
method), 360	616
	end_of_one_cycle() (pymea-
sure.instruments.siglenttechnologies.siglent_spdi	base.SPDCkane.ehstruments.temptronic.ATSBase method),

method), 547

(pymea- enabled (pymeasure.instruments.activetechnologies.AWG401x\_AFG

END 0E	616	,		property)			. 1 11	D2.470 A
END_OF	_SWEEP sure.instruments.hp.hp856Xx.StatusReg	* *	error_	status <i>property</i> )		re.instrun	nents.hp.H	P34/8A
	attribute), 303		error	status()	), 20)			(рутеа-
end_of		(рутеа-			ruments.ter	nptronic.A		
	sure.instruments.temptronic.ATSBase n			616		1		,,,
	616		ErrorC	ode	(class	iı	n	рутеа-
end_se	quence()	(рутеа-		sure.instr	ruments.hp	.hp856Xx	), 302	
	sure.instruments.advantest.advantestR6	24X.Advar	vHæsst 1860	ø∦e	(class	iı	n	рутеа-
	method), 121			sure.instr	ruments.ter	nptronic.t	emptronic_	_base),
energy	(pymeasure.instruments.thorlabs.Thorlab			620				
	property), 626		errorf	lags_to_t				(pymea-
enter_		(рутеа-			ruments.ptv	w.ptwUNI	DOS n	nethod),
	sure.instruments.temptronic.ATSBase n			514			056V II	D056V
onton	616		errors		ure.instrun	nenis.np.n	ірозолх.п	POJUAX
enter_	sure.instruments.temptronic.ATSBase n	(pymea- nethod)	errors	attribute) (pymeasu		ents newn	ort FSP30	0 prop-
	616	icinoa),	CITOIS	<i>erty</i> ), 47		chis.hewp	ori.Est 50	σ ριορ
entry_	level_strategy	(рутеа-	errors	(pymeasure	e.instrume	nts.spellm	anhv.Spell	manXRV
	sure.instruments.activetechnologies.AW	G401x_AV	WG	property)	), 565			
	property), 113		ese2	(pymeasure		ıts.anritsu	ı.AnritsuM	S9710C
	neasure.adapters.PrologixAdapter prope	• .		property)				
	measure.adapters.PrologixAdapter prope			(class in p			_	
err_st	atus (pymeasure.instruments.srs.SR830 erty), 576	prop-	esr2	(pymeasure property)		ıts.anrıtsu	ı.AnrıtsuM	S9/10C
error	(pymeasure.instruments.keithley.Keithle	v2260B	Estima	torThread		lass	in	рутеа-
01101	property), 336	,,==002			olay.widget:			
error	(pymeasure.instruments.keithley.Keith	ley2400	Estima	torWidget		lass	in	рутеа-
	property), 346			sure.disp	olay.widget:	s.estimato	r_widget),	82
error	(pymeasure.instruments.keithley.Keith	ley2450	etalon		asure.instru	ıments.acı	ulight.arga	s.Argos
error	property), 354 (pymeasure.instruments.keithley.Keith	lav2600	Furoto	<pre>property) stHPP1202</pre>		(class	in	nymaa
61101	property), 363	ie y 2000	Euroce		ruments.eu	•		рутеа-
error	(pymeasure.instruments.keithley.Keith	lev2700	Eurote	stHPP1202				tus
01101	property), 366	,_,			pymeasure			
error	(pymeasure.instruments.keithley.Keith	ley6221	evalua	te_metada				(рутеа-
	property), 373	•		sure.expe	eriment.pro	ocedure.P	rocedure n	nethod),
error	(pymeasure.instruments.keithley.Keithle	ey6517B		65				
	property), 380		event_	reg(pymea	asure.instri	uments.ro	hdeschwar	z.sfm.SFM
error	(py measure. instruments. newport. ESP 300)	) prop-		property)				
	erty), 477		event_			ure.instru	ments.hp.I	HP437B
error(	pymeasure.instruments.siglenttechnologie	_	_					
	property), 545		event_	status_er				(pymea-
error	(pymeasure.instruments.texio.TexioPSW property), 623	300L30		property)		vantest.ac	ivantest <b>k</b> o.	24X.AdvantestR624X
error_	code (pymeasure.instruments.temptronic.	ATSBase	event_			ts		(рутеа-
	property), 616				ruments.an			
ERROR_	PRESENT	(рутеа-		property)	), 239			
	sure.instruments.hp.hp856Xx.StatusReg	ister	event_	status_re	-			(рутеа-
	attribute), 303					lvantest.ac	lvantestR6	24X.AdvantestR624X
error_	reg (pymeasure.instruments.anaheimauto							
	property), 225		exchan	ge_traces		1 050		(рутеа-
error_	=	(pymea-			ruments.hp	o.hp856Xx	.нР856Хх	
	sure.instruments.advantest.advantestR6	∠4A.Advai	ntestK02	4Ametnod).	. <b>4</b> 88			

exec() (pymeasure.display.console.Manage method), 76	dConsole	sure.instruments.anritsu.AnritsuMS209 property), 233	90A
execute() (pymeasure.experiment.procedure.F method), 65	Procedure	<pre>external_enabled     sure.instruments.kuhneelectronic.Kusg</pre>	(pymea- 245
execute_analysis()	(рутеа-	property), 442	
	4 2	* * * * * * * * * * * * * * * * * * * *	(pymea-
method), 640	resinger.	sure.instruments.rohdeschwarz.sfm.SF	4.2
execute_selftest()	(рутеа-	property), 530	,,
			(mayor o a
sure.instruments.ptw.ptwDIAMENTO	Λ		(pymea-
method), 512	(	sure.instruments.rohdeschwarz.sfm.SF	VI
<pre>expand_channel_string()</pre>	(pymea-	property), 530	/
sure.instruments.ni.virtualbench.Virtu	аівепсп		(pymea-
method), 491		sure.instruments.rohdeschwarz.sfm.SF	M
expected_protocol() (in module pymeasure.		property), 530	
Experiment (class in pymeasure.display.manag	ger), 80	_	(рутеа-
Experiment (class in	рутеа-	sure.instruments.pendulum.cnt91.CNT	91
sure.experiment.experiment), 63		property), 508	
ExperimentQueue (class in	рутеа-	external_trigger_delay	(pymea-
sure.display.manager), 80		sure.instruments.anritsu.AnritsuMS464	$^{4}xB$
<pre>export_signal()</pre>	(рутеа-	property), 240	
sure.instruments.ni.virtualbench.Virtu	alBench.Di		(рутеа-
method), 479		sure.instruments.anritsu.AnritsuMS464	$^{4}xB$
ExpressionValidator (class in	рутеа-	property), 240	
sure.display.widgets.sequencer_widge	1 2	* * · * ·	(pymea-
85	- /,	sure.instruments.anritsu.AnritsuMS464	
ext_ref_base_unit	(рутеа-		ND
sure.instruments.rohdeschwarz.sfm.SI			(pymea-
property), 529	171	sure.instruments.anritsu.AnritsuMS464	* ·
ext_ref_extension	(mma a a		·xD
	(pymea-	property), 240	(O man)
sure.instruments.rohdeschwarz.sfm.SF	' IVI	extfreqency (pymeasure.instruments.srs.SR86	o prop-
property), 529	,	erty), 580	,
ext_trig_out		extract_value()	(pymea-
sure.instruments.agilent.Agilent33500	prop-	sure.instruments.keithley.Keithley6517	В
erty), 176		static method), 380	
ext_trig_out	(pymea-	_	
sure.instruments.keysight.Keysight811	160A	F	
property), 435		factory_reset()	(рутеа-
ext_vid_connector	(pymea-	sure.instruments.keysight.KeysightDSC	
sure.instruments.rohdeschwarz.sfm.SF	FM	method), 409	
property), 529		FakeAdapter (class in pymeasure.adapters), 58	
extensions (pymeasure.display.widgets.fileinp	ut widget.l	FitelkethSidsshent (class in	рутеа-
property), 82	_ 0	sure.instruments.fakes), 107	рушей
	Frequency	$R$ $\mathfrak{E}$ $E$	akaShoraA21
attribute), 302		property), 452	ikeShore421
External (pymeasure.instruments.hp.hp856Xx.	MixerMode	efact mode enabled	(mayor o a
attribute), 301	111111111111111111111111111111111111111		(pymea-
External (pymeasure.instruments.hp.hp856Xx.	Source I av	sure.instruments.advantest.advantestR6	024X.SMUCnannei
	SourceLeve	1 1 277	~
attribute), 303	T.:'M -	fault_code (pymeasure.instruments.aja.DCX	S prop-
External (pymeasure.instruments.hp.hp856Xx.	iriggermo	0.05), = 15	
attribute), 304	(	Fav (pymeasure.instruments.hp.hp856Xx.Sweep	Out at-
external_arming_start_slope	(pymea-	tribute), 304	
sure.instruments.pendulum.cnt91.CNT	191	fet (pymeasure.instruments.signalrecovery.L	DSP7225
property), 508	,	property), 553	
external_current	(pymea-		

fet	(pymeasure.instruments.signalrecovery. property), 560	DSP7265	fetch	property), 234 scan(pymeasure.instr	umonts anritsu An	ritsuMS2090A
fetch	_control	(рутеа-	IC CCII_	property), 234	umenis.anniisu.nn	1113111115207011
IC CCII	sure.instruments.anritsu.AnritsuMS20	4.2	fetch	semask		(рутеа-
	property), 233	7071	IC CCII_	sure.instruments.anr	ritsu AnritsuMS20	4.7
fetch	_density	(рутеа-		property), 234	usu.nn usum520.	70/1
Te cen	sure.instruments.anritsu.AnritsuMS20		fotch	ssb (pymeasure.instru	mants anvitsu Anv	iteuMS2000A
		790A	Tetti_		mems.am usu.Am	usum52090A
fo+ch	property), 233 _eirpower	(mum a a	fotch	property), 234		(рутеа-
reccn_	_e11 powe1 sure.instruments.anritsu.AnritsuMS20	(pymea-	recci_	.sync_evm sure.instruments.anr	ritau AnritauMS200	A >
	property), 233	190A		property), 234	usu.Anrusum5203	90A
fetch_	_eirpower_data	(pymea-	fetch_	sync_power		(pymea-
	sure.instruments.anritsu.AnritsuMS20	990A		sure.instruments.anr	itsu.AnritsuMS209	90A
	property), 233			property), 234		
fetch	_eirpower_max	(рутеа-	fetch_	tae (pymeasure.instru	ments.anritsu.Anr	itsuMS2090A
	sure.instruments.anritsu.AnritsuMS20	090A		property), 235		
	property), 233		field	(pymeasure.instrumen	ets.ami.AMI430 v	property).
fetch	emf (pymeasure.instruments.anritsu.An	ritsuMS209		223	r	F
	property), 233			 (pymeasure.instrument.	s.fwbell.FWBell50	)80 prop-
fetch	_emf_meter	(рутеа-		erty), 258	ny moonin' madeile o	oo p.op
10001	sure.instruments.anritsu.AnritsuMS20		field	(pymeasure.instrumen	ts lakeshore Lakes	Shore421
	property), 234	7071	IICIU	property), 452	is.iakesnore.Eakes	JH01C421
fotch	_emf_meter_sample	(рутеа-	fiold	(pymeasure.instrumen	ets lakashora Laka	Shore 125
Te ccii_	sure.instruments.anritsu.AnritsuMS20		iieiu	property), 454	is.iukesnore.Lukes	511016425
		790A	£; 0147	property), 434 pymeasure.instruments	a arfandinatnum ant	a IDC120 10
fo+ ob	property), 234	(	rrera (		i.oxjorainsirumeni	S.IF S120_10
retch_	_interference_power	(pymea-	£: ~1 d	property), 500	um anta labaah ana	I akaChana421
	sure.instruments.anritsu.AnritsuMS20	190A	iieia_	mode (pymeasure.instr	umenis.iakesnore.	LakeSnore421
c . 1	property), 234	,	c: 11	property), 452		
ietch_	_mimo_antenas	(pymea-	iieia_	multiplier		(pymea-
	sure.instruments.anritsu.AnritsuMS20	190A		sure.instruments.lak	eshore.LakeShore2	421
	property), 234			property), 452		
fetch_	_ocupied_bw	(рутеа-	field_	range (pymeasure.inst	ruments.lakeshore	2.LakeShore421
	sure.instruments.anritsu.AnritsuMS20	)90A		property), 452		
	property), 234		$field_{-}$	range_raw		(pymea-
fetch_	_ota_mapping	(pymea-		sure.instruments.lak	eshore.LakeShore4	421
	sure.instruments.anritsu.AnritsuMS20	090A		property), 452		
	property), 234			raw (pymeasure.instru	ments.lakeshore.L	akeShore421
fetch_	_pan (pymeasure.instruments.anritsu.An	ritsuMS209	OA	property), 452		
	property), 234		field_	setpoint		(pymea-
fetch	_pbch_constellation	(рутеа-		sure.instruments.oxfo	ordinstruments.IP:	S120_10
	sure.instruments.anritsu.AnritsuMS20			property), 500		
	property), 234		fields		uments.fwbell.FW	Bell5080
fetch	_pci (pymeasure.instruments.anritsu.An	ritsuMS209		method), 258	<b>y</b>	
	property), 234		Filame		in	рутеа-
fetch	_pdsch(pymeasure.instruments.anritsu.2	AnritsuMS2		sure.instruments.spe		1 .
IC CCII	property), 234	1111 1151111152	07011	566	umannv.speumanz	iii ( ),
fotch	_pdsch_constellation	(рутеа-	filamo	nt (pymeasure.instrum	ants spallmanhy S	nallman V RV
Te ccii_	sure.instruments.anritsu.AnritsuMS20		TTTAME	attribute), 564	enis.speiimannv.s	реининик
		790A	E:laTm	* *		
C-+-l-	property), 234			putWidget (cla		pymea-
retcn_	_peak (pymeasure.instruments.anritsu.Ai	าหนรน/ศ520		sure.display.widgets.		
C- 1	property), 234	4 . 3400			.wiagets.fileinput_	widget.FileInputWidget
retch_	_power (pymeasure.instruments.anritsu.z	AnritsuMS2		property), 83		/
	property), 234			me_base		(pymea-
fetch_	_rrm(pymeasure.instruments.anritsu.An	ritsuMS209	OA	sure.display.widgets.	fileinput_widget.F	ileInputWidget

property), 83		firmwar	e_version	(рутеа-
filename_extension	(pymea-		sure.instruments.mksinst.mks974b.M	A .
sure.display.widgets.fileinput_widget.i		idget	property), 475	
property), 83	•	-	e_version	(рутеа-
filename_fixed	(pymea-		sure.instruments.tcpowerconversion.	CXN
sure.display.widgets.fileinput_widget.i	FileInputWi	idget	property), 586	
property), 83	_	firmwar	e_version	(рутеа-
FilenameLineEdit (class in	рутеа-		sure.instruments.thyracont.smartline_	_v2.SmartlineV2
sure.display.widgets.filename_widget)	, 83		property), 630	
FilenameValidator (class in sure.display.widgets.filename_widget)	<i>pymea</i> -	fixup()	(pymeasure.display.widgets.filename_method), 83	widget.FilenameValidator
filer_synchronous (pymeasure.instruments		flags()		r widget.SequencerTreeMod
property), 580		5 0	method), 86	_ 0 1
	srs.SR830	Flattop	(pymeasure.instruments.hp.hp856Xx.\ attribute), 304	WindowType
filter (pymeasure.instruments.agilent.agilent	B1500.SMU	/FloatPa		рутеа-
property), 190			sure.experiment.parameters), 67	
filter (pymeasure.instruments.hp.HP437B)	property),	flow_un		(рутеа-
312	1 277		sure.instruments.proterial.rod4.ROD4	
filter (pymeasure.instruments.ni.virtualbench	.VirtualBer	ıch.Functi		
property), 483			41 (class in pymeasure.instruments.flu	ıke), 256
filter_advanced (pymeasure.instruments	srs.SR860			
property), 580			method), 47	•
filter_automatic_enabled	(pymea-	flush_r	ead_buffer()	(рутеа-
sure.instruments.hp.HP437B property	), 312		sure.adapters.FakeAdapter method),	59
filter_count	(pymea-	flush_r	ead_buffer()	(рутеа-
sure.instruments.keithley.Keithley240	0 prop-		sure.adapters.PrologixAdapter	method),
erty), 346			55	
<pre>filter_slope (pymeasure.instruments.srs.SR</pre>	830 prop-	flush_r	ead_buffer()	(рутеа-
erty), 576			sure.adapters.ProtocolAdapter	method),
<pre>filter_slope (pymeasure.instruments.srs.SR</pre>	860 prop-		58	
erty), 580		flush_r	ead_buffer()	(рутеа-
filter_state	(pymea-		sure.adapters.SerialAdapter method).	, 52
sure.instruments.keithley.Keithley240	0 prop-	flush_r	ead_buffer()	(рутеа-
erty), 346			$sure. adapters. \textit{VISAA} dapter\ method),$	
filter_synchronous	(pymea-	fm_devi	1.7	.HP8657B
sure.instruments.srs.SR830 property),	576		property), 306	
filter_synchronous	(pymea-	fm_sour	ce (pymeasure.instruments.hp.HP865	57B prop-
sure.instruments.srs.SR860 property),			erty), 306	
${\tt filter\_type} \ (pymeasure.instruments.keithley.$	<i>Keithley240</i>	0 <b>£</b> oldbac		(рутеа-
property), 346			sure.instruments.tdk.tdk_gen40_38.T	DK_Gen40_38
filter_type (pymeasure.instruments.srs.SR5	70 prop-	6 7 11	property), 590	
erty), 574	,	foldbac		(pymea-
find_img_index()	(pymea-		sure.instruments.tdk.tdk_gen80_65.T	DK_Gen80_63
sure.display.curves.ResultsImage	method),	C 1 11	property), 595	,
77	VA D	тотарас	k_enabled	(pymea-
firmware (pymeasure.instruments.ipgphotonic	s.yar.YAK		sure.instruments.tdk.tdk_gen40_38.Ta	DK_Gen40_38
property), 322	(123	£0141	property), 590	(myma a a
firmware_revision	(pymea-	тотарас	k_enabled	(pymea-
sure.instruments.hp.hp856Xx.HP856X	Xx at-		sure.instruments.tdk.tdk_gen80_65.Td	DN_Genou_03
tribute), 280	(123	£0141	property), 595	(myma a a
firmware_version  Sura instruments inficon sam 160 SOM	(pymea- 1160	τοταράς	k_reset()	(pymea- DK Gan40, 38
sure.instruments.inficon.sqm160.SQM property), 320	100		sure.instruments.tdk.tdk_gen40_38.Tamethod), 591	DK_Gen40_30
DIODCIVYIS 220			11001100011, 011	

<pre>foldback_reset()           sure.instruments.tdk.tdk_gen80_65.TD</pre>	(pymea- OK_Gen80_	property), 442 _ <b>&amp;req_stop</b> (pymeasure.instruments.rohdeschwarz.fsseries.FSSeries
method), 595		property), 540
force() (pymeasure.instruments.agilent.agilent method), 190	tB1500.SMU	Ufreq_sweep() (pymea- sure.instruments.agilent.AgilentE4980
force_gnd() (pymeasure.instruments.agilent.agi	-	0.AgilentBhhhhod), 154 frequencies (pymeasure.instruments.agilent.Agilent8722ES
<pre>force_gnd() (pymeasure.instruments.agilent.agilen</pre>		
method), 190		frequencies (pymeasure.instruments.agilent.AgilentE4408B
${\tt FORCE\_SIDE}\ (pymeasure.instruments.agilent.ag$		
attribute), 195		$frequencies ({\it pymeasure.instruments.agilent.agilentE5062A.VNAC hannel to the control of the $
<pre>force_trigger()</pre>	(pymea-	property), 159
sure.instruments.ni.virtualbench.Virtu	alBench.Mi.	ix <b>txlSqywnlClse:</b> lloscofpeymeasure.instruments.hp.HP8753E
method), 486		property), 308
<pre>force_trigger()</pre>	(рутеа-	$frequency (\it pymeasure.instruments.active technologies. AWG 401x. Channel and the control of t$
sure.instruments.teledyne.TeledyneMA	UI	property), 114
method), 611		frequency (pymeasure.instruments.agilent.Agilent33220A
${\tt format}\ ({\it pymeasure.instruments.pendulum.cnt9}$	1.CNT91	property), 173
property), 508		frequency (pymeasure.instruments.agilent.Agilent33500
${\tt format()} \ (pymeasure.display.widgets.log\_widg$		
method), 84		frequency (pymeasure.instruments.agilent.agilent33500.Agilent33500Cha
<pre>format() (pymeasure.experiment.results.CSVI</pre>		property), 179
method), 72		frequency (pymeasure.instruments.agilent.Agilent33521A
format() (pymeasure.experiment.result		property), 178
method), 72		frequency (pymeasure.instruments.agilent.Agilent34450A
fpga (pymeasure.instruments.spellmanhv.Spell		property), 163
property), 565		frequency (pymeasure.instruments.agilent.Agilent4284A
Fpu60 (class in pymeasure.instruments.novanta)		property), 198
frame (pymeasure.instruments.fakes.SwissA property), 107	ArmyFake	frequency (pymeasure.instruments.agilent.agilent4284A.Agilent4284ASperproperty), 200
<pre>frame_format</pre>		frequency (pymeasure.instruments.agilent.Agilent8257D
sure. instruments. fakes. Swiss Army Fake		property), 150
erty), 107		frequency (pymeasure.instruments.agilent.AgilentE4980
<pre>frame_height</pre>	(рутеа-	property), 154
erty), 107		frequency (pymeasure.instruments.ametek.Ametek7270 property), 221
frame_width (pymeasure.instruments.fakes.Swi property), 107	issArmyFak	k&requency (pymeasure.instruments.anapico.APSIN12G property), 227
	gerMode	frequency (pymeasure.instruments.andeenhagerling.AH2700A property), 228
<pre>free_memory_slots     sure.instruments.keysight.keysight8116</pre>		frequency (pymeasure.instruments.anritsu.AnritsuMG3692C
property), 440		frequency (pymeasure.instruments.attocube.anc300.Axis
freerun_enabled	(рутеа-	property), 247
		frequency (pymeasure.instruments.hp.HP33120A prop-
506	· · · · · · · · · · · · · · · · · · ·	erty), 262
	warz.fsserie	efressuricsy (pymeasure.instruments.hp.HP437B prop-
property), 540	<b>U</b> J	erty), 312
	ırz.fsseries.İ	Historium (pymeasure.instruments.hp.HP8116A prop-
property), 540	.,	erty), 272
	arz.fsseries	s <b>FESSprincy</b> (pymeasure.instruments.hp.hp856Xx.DemodulationMode
property), 540		attribute), 301
<pre>freq_steps_fine_enabled</pre>		frequency (pymeasure.instruments.hp.HP8657B prop-
sure.instruments.kuhneelectronic.Kuss	245 250A	erty), 306

frequency (pymeasure.instruments.inficon.sqm160.Sensor	· <b>Chreque</b> ncy_current_auto_range	(рутеа-
property), 321	sure.instruments.agilent.Agilent34450a	
frequency (pymeasure.instruments.keithley.Keithley2000	property), 163	
property), 325	frequency_current_range	(рутеа-
frequency (pymeasure.instruments.keithley.KeithleyDMM		= :
property), 387	property), 164	
frequency (pymeasure.instruments.keysight.Keysight8116		(рутеа-
property), 435	sure.instruments.anritsu.anritsuMS464	
frequency (pymeasure.instruments.rigol.rigol_dg800.Volt		
property), 524	frequency_digits	(рутеа-
frequency (pymeasure.instruments.rohdeschwarz.sfm.SFM		
property), 530	erty), 325	1 1
frequency (pymeasure.instruments.rohdeschwarz.sfm.Sou	* *	(рутеа-
property), 537	sure.instruments.keithley.KeithleyDMN	~ -
frequency (pymeasure.instruments.signalrecovery.DSP72		
property), 553	frequency_display_enabled	(рутеа-
frequency (pymeasure.instruments.signalrecovery.DSP72		4.
property), 560	tribute), 283	
frequency (pymeasure.instruments.srs.SR510 property),		(рутеа-
573	sure.instruments.kuhneelectronic.Kusg	~ -
frequency (pymeasure.instruments.srs.SR830 property),	property), 442	
576	frequency_max	(рутеа-
frequency (pymeasure.instruments.srs.SR860 property),	sure.instruments.activetechnologies.AV	
580	property), 115	
frequency (pymeasure.instruments.tcpowerconversion.CX		(рутеа-
property), 586	sure.instruments.activetechnologies.AV	
frequency (pymeasure.instruments.tektronix.afg3152c.AF		
property), 599	frequency_mode	(рутеа-
frequency (pymeasure.instruments.teledyne.teledyneT3AF		~ -
property), 602	property), 530	
	frequency_offset	(рутеа-
sure.instruments.keithley.Keithley2000 prop-	sure.instruments.anritsu.AnritsuMS209	
erty), 325	property), 235	
· · · · · · · · · · · · · · · · · · ·	frequency_offset	(рутеа-
sure.instruments.keithley.KeithleyDMM6500	sure.instruments.hp.hp856Xx.HP856X.	x at-
property), 387	tribute), 282	
	frequency_points	(рутеа-
sure.instruments.agilent.Agilent34450A	sure.instruments.agilent.AgilentE4408.	B
property), 163	property), 153	
frequency_center (pymea-	frequency_reference	(рутеа-
sure.instruments.anritsu.AnritsuMS2090A	sure.instruments.keithley.Keithley2000	prop-
property), 235	erty), 325	
frequency_center (pymea-	***	(рутеа-
	rementChasuused.instruments.hp.hp856Xx.HP856X.	x at-
property), 243	tribute), 283	
frequency_coarse (pymea-	frequency_relative	(рутеа-
sure.instruments.kuhneelectronic.Kusg245_250A		• •
property), 442	property), 388	
frequency_counter_mode_enabled (pymea-	frequency_relative_enabled	(рутеа-
sure.instruments.hp.hp856Xx.HP856Xx at-	sure.instruments.keithley.KeithleyDMN	• •
tribute), 279	property), 388	
frequency_counter_resolution (pymea-	frequency_span	(рутеа-
sure.instruments.hp.hp856Xx.HP856Xx at-	sure.instruments.anritsu.AnritsuMS209	* *
tribute) 279	property), 235	

frequency_span (pymea-	<pre>front_panel (pymeasure.instruments.srs.SR860 prop-</pre>
sure.instruments.anritsu.anritsuMS464xB.Measu	
property), 243	front_panel_brightness (pymea-
frequency_span_full (pymea-	sure.instruments.lakeshore.LakeShore421
sure.instruments.anritsu.AnritsuMS2090A	property), 452
property), 235	front_panel_display (pymea-
frequency_span_last (pymea-	sure.instruments.oxfordinstruments.ITC503
sure.instruments.anritsu.AnritsuMS2090A	property), 495
property), 235	front_panel_locked (pymea-
frequency_start (pymea-	sure.instruments.lakeshore.LakeShore421
sure.instruments.anritsu.AnritsuMS2090A	property), 452
property), 235	FSL (class in pymea-
frequency_start (pymea-	sure.instruments.rohdeschwarz.fsseries),
sure.instruments.anritsu.anritsuMS464xB.Measu	
property), 243	FSSeries (class in pymea-
frequency_step (pymea-	sure.instruments.rohdeschwarz.fsseries),
sure.instruments.agilent.AgilentE4408B	540
property), 153	FSW (class in pymea-
frequency_step (pymea-	sure.instruments.rohdeschwarz.fsseries),
sure.instruments.anritsu.AnritsuMS2090A	541
property), 235	function (pymeasure.instruments.agilent.AgilentB2985
frequency_stop (pymea-	property), 209
sure.instruments.anritsu.AnritsuMS2090A	function_(pymeasure.instruments.hp.HP34401A prop-
property), 235	erty), 264
	function_mode (pymea-
	rementChasumed.instruments.keithley.Keithley2281S
property), 244	property), 339
	fw (pymeasure.instruments.hp.HP8753E property), 308
sure.instruments.keithley.Keithley2000 prop-	<pre>fw_version(pymeasure.instruments.siglenttechnologies.siglent_spdbase.S</pre>
erty), 325	property), 545
frequency_threshold (pymea-	FWBell5080 (class in pymeasure.instruments.fwbell),
sure.instruments.keithley.KeithleyDMM6500	257
property), 388	
frequency_threshold_auto_enabled (pymea-	G
sure.instruments.keithley.KeithleyDMM6500	gain (pymeasure.instruments.signalrecovery.DSP7225
property), 388	property), 553
frequency_voltage_auto_range (pymea-	gain (pymeasure.instruments.signalrecovery.DSP7265
sure.instruments.agilent.Agilent34450A	property), 560
property), 164	gain_mode (pymeasure.instruments.srs.SR570 property),
frequency_voltage_range (pymea-	574
sure.instruments.agilent.Agilent34450A	gasflow (pymeasure.instruments.oxfordinstruments.ITC503
property), 164	property), 495
frequencypreset1 (pymeasure.instruments.srs.SR860	gasflow_configuration_parameter (pymea-
property), 580	sure.instruments.oxfordinstruments.ITC503
frequencypreset2 (pymeasure.instruments.srs.SR860	property), 495
property), 580	gasflow_control_status (pymea-
frequencypreset3 (pymeasure.instruments.srs.SR860	sure.instruments.oxfordinstruments.ITC503
property), 581	property), 495
frequencypreset4 (pymeasure.instruments.srs.SR860	<pre>gate_enabled (pymeasure.instruments.philips.PM6669</pre>
property), 581	property), 506
FrequencyReference (class in pymea-	<pre>gate_time (pymeasure.instruments.hp.HP34401A prop-</pre>
sure.instruments.hp.hp856Xx), 302	erty), 264
front_blanked (pymeasure.instruments.srs.SR570	${\tt gate\_time} \ (pymeasure.instruments.pendulum.cnt91.CNT91$
property), 574	property), 508

gen_measurement()	(рутеа-	method), 333	
sure.experiment.procedure.Procedure			(рутеа-
65	,,,	sure.instruments.keysight.Keysight811	
GeneralError (class in	рутеа-	static method), 435	
sure.instruments.newport.esp300), 47			(pymea-
Generator (class in pymeasure.generator), 60	O	sure.instruments.keysight.KeysightE36	
get() (pymeasure.instruments.agilent.agilentB.	1500 Custo		551271
	1300.Cusio	get_channels()	(mym a a
<pre>class method), 194 get_alarm_status()</pre>	(mum a a	sure.instruments.keysight.KeysightE36	(pymea-
sure.instruments.lakeshore.LakeShore	(pymea-	static method), 426	531A
	211	· · · · · · · · · · · · · · · · · · ·	(mym a a
method), 446	(	get_channels()	(pymea-
get_analysis()	(pymea-	sure.instruments.lecroy.LeCroyT3DSC	71204
sure.instruments.yokogawa.aq6370ser	ies.AQ057		9 4156 1 9 4156
method), 640		<pre>get_data() (pymeasure.instruments.agilent.ag</pre>	ilent4136.Agilent4136
get_array() (in module	рутеа-	method), 167	
sure.experiment.experiment), 64		get_data() (pymeasure.instruments.agilent.Ag	ilent4294A
get_array_steps() (in module	рутеа-	method), 171	
sure.experiment.experiment), 64		<pre>get_descriptor()</pre>	(pymea-
get_array_zero() (in module	рутеа-	sure.instruments.siglenttechnologies.si	iglent_sds1072cml.Voltage <b>(</b>
sure.experiment.experiment), 64		method), 549	
get_buffer()	(рутеа-	<pre>get_error_message()</pre>	(рутеа-
sure.instruments.signal recovery.DSP7	7225	sure.instruments.agilent.AgilentE5270	$\partial B$
method), 554		method), 214	
get_buffer()	(рутеа-	<pre>get_estimates()</pre>	(pymea-
sure.instruments.signalrecovery.DSP7	265	sure.display.widgets.estimator_widget	.EstimatorWidget
method), 560		method), 82	
<pre>get_buffer()</pre>	srs.SR830	<pre>get_estimates()</pre>	(рутеа-
method), 576		sure.experiment.procedure.Procedure	
<pre>get_calibration_information()</pre>	(рутеа-	65	,,
sure.instruments.ni.virtualbench.Virtu		<pre>get_filename()</pre>	(рутеа-
method), 491		sure.display.console.ManagedConsole	4.
<pre>get_channel_pairs()</pre>	(рутеа-	method), 76	
sure.instruments.common_base.Comn			(pymea-
static method), 101	ionBuse	sure.instruments.ni.virtualbench.Virtu	
get_channel_pairs()	(рутеа-	method), 491	arBenen
sure.instruments.keithley.Keithley2200			(рутеа-
method), 333	) siuit	sure.instruments.srs.SR860 property),	* *
**	(nymaa		
		get_operation_times()	
sure.instruments.keysight.Keysight811	OUA	sure.instruments.novanta.Fpu60 492	method),
static method), 435	(		(
<pre>get_channel_pairs()</pre>	(pymea-	<pre>get_power_bandwidth()</pre>	(pymea-
sure.instruments.keysight.KeysightE36	0312A	sure.instruments.hp.hp856Xx.HP856X	X
static method), 418	,	method), 285	,
get_channel_pairs()	(pymea-	<pre>get_procedure()</pre>	(pymea-
sure.instruments.keysight.KeysightE36	531A	sure.display.widgets.inputs_widget.Inp	outsWidget
static method), 426		method), 84	
get_channel_pairs()	(рутеа-	<pre>get_relay_mode()</pre>	(pymea-
sure.instruments.lecroy.LeCroyT3DSC	01204	sure.instruments.lakeshore.LakeShore	211
static method), 462		method), 446	
get_channels()	(рутеа-	<pre>get_scaling() (pymeasure.instruments.s</pre>	rs.SR830
sure.instruments.common_base.Comn	10nBase	method), 576	
static method), 101		<pre>get_sensor_transition()</pre>	(рутеа-
get_channels()	(pymea-	$sure.instruments.thyracont.smartline\_$	v2.SmartlineV2
sure instruments keithley Keithley2200	) static	method) 630	

get_sig	nal_strength_indicator	(рутеа-	tribute), 280	
	sure.instruments.srs.SR860 propert		grid_display	(pymea-
get_sta	ate_of_channels()	(pymea-	sure.instruments.lecroy.LeCroyT3D	SO1204
	sure.instruments.keithley.Keithley27	/00	property), 462	(
	method), 366	(	ground_all()	(pymea-
get_tra	ace_data_a()	(pymea-	sure.instruments.attocube.anc300.A	NC300Controlle
	sure.instruments.hp.hp856Xx.HP85 method), 286	OAX	method), 246 group_trigger_mode	(pymea-
net tra	nce_data_b()	(рутеа-	sure.instruments.hp.HP437B proper	
gcc_cr	sure.instruments.hp.hp856Xx.HP85		sure.instruments.np.111 +3/D proper	19), 312
	method), 286	0120	Н	
get_tri	.gger_config()	(рутеа-	handle_abort()	(рутеа-
_	sure.instruments.siglenttechnologie.	s.siglent_sds1	1072cml.Trigger.Experienent.workers.Worker	method),
	method), 550		71	memou),
get_wav	veform()	(pymea-	handle_batch_record()	(pymea-
	sure.instruments.siglenttechnologie.	s.siglent_sds1	1072cml.Voltage.Chgennelent.workers.Worker	method),
	method), 549		71	,,
get_wl_		(pymea-	handle_deprecated_host_arg()	(pymea-
	sure.instruments.keysight.Keysight?	<i>N7776C</i>	sure.instruments.attocube.anc300.A	NC300Controlle
	method), 412		method), 246	
get_xda	ata() (pymeasure.instruments.yokogo	awa.aq6370so	eriand 2637 Prories	(pymea-
_	method), 641		sure.experiment.workers.Worker	method),
get_yda	ata() (pymeasure.instruments.yokogo	awa.aq6370si	eries.AQ637 <sub>4</sub> Series	
	method), 641	70 110	handle_record()	(pymea-
	(in module pymeasure.instruments.c		sure.experiment.workers.Worker	method),
	(in module pymeasure.instruments.c		71	
gettime	ebase (pymeasure.instruments.srs.S. erty), 581	K80U prop-	Hanning (pymeasure.instruments.hp.hp856Xx attribute), 304	:.WindowType
gpib()	(pymeasure.adapters.PrologixAdapters.	er method),	hardcopy_setup()	(pymea-
	55	,	sure.instruments.teledyne.TeledyneM	<i>IAUI</i>
gpib_ac		(pymea-	method), 611	
	sure.instruments.rohdeschwarz.sfm.	SFM	hardcopy_setup_current	(рутеа-
	property), 530	(	sure.instruments.teledyne.TeledyneN	<i>IAUI</i>
gprb_re	ead_timeout sure.adapters.PrologixAdapter	(pymea-	property), 611	,
	55	property),	hardware_version	(pymea-
anih sa	oftware_version	(рутеа-	sure.instruments.mksinst.mks974b.M	1KS9/4B
	sure.instruments.racal.Racal1992	* *	property), 475	A4 -1-7270
	519	property),	harmonic (pymeasure.instruments.ametek.A	Ametek/2/0
GPIB tr	rigger() (pymeasure.instruments.h	p.HP8116A	property), 221 harmonic (pymeasure.instruments.signalreco	vary DCD7225
	method), 271	F	property), 554	very.DSI 7225
GPIB_tr	rigger()	(рутеа-	harmonic (pymeasure.instruments.signalreco	very DSP7265
	sure.instruments.hp.HPLegacyInstr		property), 561	ver y.DS1 7203
	method), 316		harmonic (pymeasure.instruments.srs.SR830	) property)
gps (	pymeasure.instruments.anritsu.Anrit	suMS2090A	577	p.ope,,,
	property), 235		harmonic (pymeasure.instruments.srs.SR860	property).
gps_all	(pymeasure.instruments.anritsu.Anr	itsuMS2090A	581	F F / / /
	property), 235		harmonic_number_lock	(pymea-
gps_ful	.1 (pymeasure.instruments.anritsu.An	ritsuMS2090		property),
	property), 235		298	
gps_las	st (pymeasure.instruments.anritsu.Ar	ritsuMS2090	Aharmonic_number_lock_enabled	(pymea-
	property), 235		sure.instruments.hp.HP8561B	property),
graticu	le_enabled	(pymea-	299	
	sure.instruments.hp.hp856Xx.HP85	6Xx at-		

harmonicdual (pymeasure.instruments.srs.SR860 property), 581	sure.instruments.agilent.Agilent4284A prop-
has_amplitude_modulation (pymea-	erty), 198
	HMP4040 (class in pymea-
erty), 150	sure.instruments.rohdeschwarz.hmp), 542
has_modulation (pymea-	hold() (pymeasure.instruments.hp.hp856Xx.HP856Xx
sure.instruments.agilent.Agilent8257D prop-	method), 280
erty), 150	hold() (pymeasure.instruments.ptw.ptwUNIDOS
has_next() (pymeasure.display.manager.ExperimentQuei	
method), 80	hold_function (pymea-
has_persistent_switch_enabled() (pymea-	sure.instruments.anritsu.anritsuMS464xB.MeasurementChannel
sure.instruments.ami.AMI430 method), 223	property), 244
	hold_function_all_channels (pymea-
sure.instruments.agilent.Agilent8257D prop-	sure.instruments.anritsu.AnritsuMS464xB
erty), 150	property), 240
= :	hold_time (pymeasure.instruments.agilent.agilent4156.Agilent4156
sure.instruments.hp.HP8116A property),	property), 167
272	home() (pymeasure.instruments.anaheimautomation.DPSeriesMotorContro
head (pymeasure.instruments.temptronic.ATSBase prop-	method), 225
erty), 616	home() (pymeasure.instruments.newport.esp300.Axis
head_temperature (pymea-	method), 477
sure.instruments.novanta.Fpu60 property),	horizontal_time_div (pymea-
492	sure.instruments.srs.SR860 property), 581
header() (pymeasure.experiment.results.Results method), 72	hotcathode (pymeasure.instruments.thyracont.smartline_v2.VSH attribute), 632
headerData() (pymea-	HP11713A (class in pymeasure.instruments.hp), 309
sure.display.widgets.sequencer_widget.Sequencer	r <b>The 3 MADA</b> (class in pymeasure.instruments.hp), 262
method), 86	HP3437A (class in pymeasure.instruments.hp), 266
headerData() (pymea-	HP3437A. SRQ (class in pymeasure.instruments.hp), 266
sure.display.widgets.table_widget.PandasModelB	AMP34401A (class in pymeasure.instruments.hp), 263
method), 89	HP3478A (class in pymeasure.instruments.hp), 268
Heater (class in pymea-	HP3478A.ERRORS (class in pymeasure.instruments.hp),
sure.instruments.oxfordinstruments.mercuryitc),	268
503	HP437B (class in pymeasure.instruments.hp), 311
heater (pymeasure.instruments.oxfordinstruments.ITC503	
property), 496	HP6632A.ERRORS (class in pymeasure.instruments.hp),
heater_gas_mode (pymea-	317
sure.instruments.oxfordinstruments.ITC503	HP6632A.ST_ERRORS (class in pymea-
property), 496	sure.instruments.hp), 317
heater_voltage (pymea-	HP6633A (class in pymeasure.instruments.hp), 319
sure.instruments.oxfordinstruments.ITC503	HP6634A (class in pymeasure.instruments.hp), 319
property), 496	HP8116A (class in pymeasure.instruments.hp), 271
High (pymeasure.instruments.hp.hp856Xx.PeakSearchModattribute), 302	
	HP8116A.Direction (class in pymea- sure.instruments.hp), 271
	HP8560A (class in pymeasure.instruments.hp), 296
sure.instruments.rohdeschwarz.sfm.SFM	HP8561B (class in pymeasure.instruments.hp), 298
property), 530	HP8657B (class in pymeasure.instruments.hp), 305
high_impedance (pymea-	
sure.instruments.rigol.rigol_dg800.VoltageChan	` ***
property), 525	HP8753E (class in pymeasure.instruments.hp), 307
high_level (pymeasure.instruments.hp.HP8116A prop-	
erty), 272	sure.instruments.hp), 316

HRADC (pymeasure.instruments.agilent.agilentB1500.ADC) attribute), 194	Typole (pymeasure.instruments.keithley.Keithley2281S prop- erty), 339
HSADC (pymeasure.instruments.agilent.agilentB1500.ADC) attribute), 194	Tyjpæ (pymeasure.instruments.keithley.Keithley2306 prop- erty), 342
HSADC_PULSED (pymea-	id (pymeasure.instruments.keithley.Keithley2400 prop-
sure.instruments.agilent.agilentB1500.ADCType attribute), 195	erty), 346 id (pymeasure.instruments.keithley.Keithley2450 prop-
HTMLFormatter (class in pymea-	erty), 354
$sure. display. widgets. log\_widget),  84 \\ \text{HTR\_MB} \ (pymeasure. instruments. oxfordinstruments. Mercure and the state of the stat$	id (pymeasure.instruments.keithley.Keithley2510 prop- yiTC erty), 360
attribute), 502 humidity (pymeasure.instruments.agilent.AgilentB2985	id (pymeasure.instruments.keithley.Keithley2600 property), 363
property), 209	id (pymeasure.instruments.keithley.Keithley2700 prop-
$\verb hv_on_timer  (pymeasure. instruments. spellmanhv. $	
property), 565	id (pymeasure.instruments.keithley.Keithley2750 property), 369
I IBeamSmart (class in pymea-	id (pymeasure.instruments.keithley.Keithley4200 property), 405
sure.instruments.toptica.ibeamsmart), 633	id (pymeasure.instruments.keithley.Keithley6221 prop-
id (pymeasure.instruments.advantest.advantestR3767CG.A	AdvantestR3967C673
property), 117	id (pymeasure.instruments.keithley.Keithley6517B prop-
id (pymeasure.instruments.aja.DCXS property), 219	erty), 380
id (pymeasure.instruments.ametek.Ametek7270 property), 221	id (pymeasure.instruments.keithley.KeithleyDAQ6510 property), 401
id (pymeasure.instruments.andeenhagerling.AH2700A property), 228	id (pymeasure.instruments.keysight.Keysight81160A property), 436
id (pymeasure.instruments.danfysik.Danfysik8500 property), 250	id (pymeasure.instruments.keysight.KeysightE36312A property), 418
id (pymeasure.instruments.eurotest.EurotestHPP120256 property), 255	id (pymeasure.instruments.keysight.KeysightE3631A property), 426
id (pymeasure.instruments.fluke.Fluke7341 property), 256	id (pymeasure.instruments.lecroy.LeCroyT3DSO1204 property), 462
id (pymeasure.instruments.fwbell.FWBell5080 property), 258	id (pymeasure.instruments.philips.PM6669 property), 506
id (pymeasure.instruments.generic_types.SCPIMixin property), 108	id (pymeasure.instruments.proterial.ROD4 property), 510
id (pymeasure.instruments.heidenhain.ND287 property), 260	id (pymeasure.instruments.ptw.ptwDIAMENTOR property), 513
id (pymeasure.instruments.hp.HP6632A property), 318	id (pymeasure.instruments.ptw.ptwUNIDOS property),
id (pymeasure.instruments.hp.hp856Xx.HP856Xx at-	514 id (pymeasure.instruments.signalrecovery.DSP7225
tribute), 280	property), 554
id (pymeasure.instruments.hp.HP8657B attribute), 306 id (pymeasure.instruments.hp.HP8753E property), 308	id (pymeasure.instruments.signalrecovery.DSP7265
id (pymeasure.instruments.Instrument property), 104	property), 561
id (pymeasure.instruments.ipgphotonics.yar.YAR prop-	id (pymeasure.instruments.srs.SR830 property), 577
erty), 322	id (pymeasure.instruments.tcpowerconversion.CXN
id (pymeasure.instruments.keithley.Keithley2000 prop-	property), 586
erty), 326	<pre>id(pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38</pre>
id (pymeasure.instruments.keithley.Keithley2182 prop-	property), 591
erty), 396	$\verb"id" (pymeasure.instruments.tdk.tdk\_gen80\_65.TDK\_Gen80\_65$
${\tt id}\ (\textit{pymeasure.instruments.keithley}. \textit{Keithley} 2200\ \textit{prop-}$	property), 595
erty), 333	id (pymeasure.instruments.teledyne.TeledyneT3AFG
id (pymeasure.instruments.keithley.Keithley2260B property), 336	property), 600 id (pymeasure.instruments.texio.TexioPSW360L30 prop-

erty), 623	sure. instruments. advantest. advantest R624X. Advantest R624X
${\tt id}\ (pymeasure. instruments. thermotron. Thermotron 380$	0 method), 120
property), 625	init_spa_self (pymea-
id (pymeasure.instruments.yokogawa.Yokogawa765	1 sure.instruments.anritsu.AnritsuMS2090A
property), 638	property), 235
$\verb"id()" (pymeasure.instruments.mksinst.mks974b.MKS97$	(B init_sweep() (pymea-
method), 475	sure.instruments.anritsu.AnritsuMS2090A
${\tt identity} (py measure. instruments. ox for dinstruments. M$	ercuryiTC method), 235
property), 502	<pre>init_transient()</pre>
IF_bandwidth (pymed	
sure.instruments.agilent.agilentE5062A.VNAC	Channel method), 209
property), 158	<pre>init_trigger()</pre>
IFBW (pymeasure.instruments.hp.HP8753E property 307	), sure.instruments.hp.HP34401A method), 264
ImageFrame (class in pymed	<i>a</i> - initalize_oven() ( <i>pymea</i> -
sure.display.widgets.image_frame), 83	sure.instruments.thermotron.Thermotron3800
ImageWidget (class in pymed	a- method), 625
sure.display.widgets.image_widget), 83	<pre>initialize_all_smus()</pre>
imode (pymeasure.instruments.signalrecovery.DSP722 property), 554	5 sure.instruments.agilent.agilentB1500.AgilentB1500 method), 185
imode (pymeasure.instruments.signalrecovery.DSP726	
property), 561	sure.instruments.agilent.agilentB1500.AgilentB1500
impedance (pymeasure.instruments.agilent.AgilentE498	
property), 154	<pre>initiate_sweep()</pre>
impedance (pymeasure.instruments.tektronix.afg3152c.	AFG3152CChauraehstruments.yokogawa.aq6370series.AQ6370Series
property), 599	method), 641
impedance_mode (pymed	a- Input (class in pymeasure.display.inputs), 77
sure.instruments.agilent.Agilent4284A property), 198	o- input_0 (pymeasure.instruments.lakeshore.LakeShore224 attribute), 447
impedance_range (pymed	
sure.instruments.agilent.Agilent4284A prop	- · · · · · · · · · · · · · · · · · · ·
erty), 199	<pre>input_A(pymeasure.instruments.lakeshore.LakeShore331</pre>
<pre>index() (pymeasure.display.widgets.sequencer_widget.</pre>	
method), 86	input_A(pymeasure.instruments.lakeshore.LakeShore3xx
information (pymeasure.instruments.hcp.TC038 prop	o- attribute), 449
erty), 260	<pre>input_B (pymeasure.instruments.lakeshore.LakeShore224</pre>
init() (pymeasure.instruments.agilent.AgilentB298	attribute), 447
method), 204	$input\_B$ (pymeasure.instruments.lakeshore.LakeShore331
<pre>init_acquisition()</pre>	a-attribute), 448
sure.instruments.agilent.AgilentB2981	$\verb"input_B" (pymeasure.instruments.lakeshore.LakeShore 3xx$
method), 204	attribute), 449
<pre>init_all_sweep()</pre> <pre>(pymee)</pre>	a- input_C (pymeasure.instruments.lakeshore.LakeShore3xx
sure.instruments.anritsu.AnritsuMS2090A	attribute), 449
method), 235	<pre>input_C1(pymeasure.instruments.lakeshore.LakeShore224</pre>
init_continuous (pymed	
sure.instruments.anritsu.AnritsuMS2090A	input_C1 (pymeasure.instruments.lakeshore.LakeShore3xx
property), 235	attribute), 449
init_curve_buffer() (pymed	· 4.
sure.instruments.signalrecovery.DSP7225	attribute), 447
method), 554	input_C2 (pymeasure.instruments.lakeshore.LakeShore3xx
init_curve_buffer() (pymed	
sure.instruments.signalrecovery.DSP7265	input_C3 (pymeasure.instruments.lakeshore.LakeShore224
method), 561 init sequence() (nymethod)	attribute), 447

attribute), 449	property), 581
input_C4( <i>pymeasure.instruments.lakeshore.LakeShore224</i> input_	_signal (pymeasure.instruments.srs.SR860 prop-
attribute), 447	erty), 582
input_C4( <i>pymeasure.instruments.lakeshore.LakeShore3xx</i>	_voltage_mode
attribute), 450	sure.instruments.srs.SR860 property), 582
input_C5 (pymeasure.instruments.lakeshore.LakeShore224Inputs	sWidget (class in pymea-
attribute), 447	sure.display.widgets.inputs_widget), 84
input_config (pymeasure.instruments.srs.SR830 prop- inser	t_id() (pymeasure.instruments.advantest.advantestR624X.SMUCha
erty), 577	method), 127
***	t_id() (pymeasure.instruments.attocube.anc300.Axis
erty), 581	method), 247
•	t_id() (pymeasure.instruments.Channel method),
property), 577	106
	t_id() (pymeasure.instruments.keithley.keithley2200.PSChannel
property), 581	method), 334
	t_id() (pymeasure.instruments.tektronix.afg3152c.AFG3152CChan
sure.instruments.srs.SR860 property), 581	method), 599
	t_id() (pymeasure.instruments.teledyne.teledyne_oscilloscope.Telea
attribute), 450	method), 608
input_D1 (pymeasure.instruments.lakeshore.LakeShore224instat	
	= :
attribute), 447	sure.instruments.razorbill.razorbillRP100
input_D1 (pymeasure.instruments.lakeshore.LakeShore3xx	property), 521
	nt_voltage_2 (pymea-
input_D2 (pymeasure.instruments.lakeshore.LakeShore224	sure.instruments.razorbill.razorbillRP100
attribute), 447	property), 521
input_D2 (pymeasure.instruments.lakeshore.LakeShore3xxinstan	
attribute), 450	method), 60
input_D3 (pymeasure.instruments.lakeshore.LakeShore224Instru	
	erInput (class in pymeasure.display.inputs), 77
input_D3 (pymeasure.instruments.lakeshore.LakeShore3xxIntego	_ ·
attribute), 450	sure.experiment.parameters), 68
input_D4( <i>pymeasure.instruments.lakeshore.LakeShore224</i> integ	
attribute), 447	sure.instruments.oxfordinstruments.ITC503
input_D4 (pymeasure.instruments.lakeshore.LakeShore3xx	property), 496
	ration_time ( <i>pymea-</i>
input_D5 (pymeasure.instruments.lakeshore.LakeShore224	sure.instruments.agilent.agilent4156.Agilent4156
attribute), 448	property), 168
input_D5 ( <i>pymeasure.instruments.lakeshore.LakeShore3xx</i> integr	ration_time (pymea-
attribute), 450	sure.instruments.ptw.ptwUNIDOS property),
input_enabled (pymea-	514
sure.instruments.agilent.AgilentB2981 prop-intens	sity (pymeasure.instruments.lecroy.LeCroyT3DSO1204
erty), 204	property), 462
input_enabled (pymea- intens	sity (pymeasure.instruments.teledyne.TeledyneOscilloscope
sure.instruments.aimtti.ld400p.LD400P	property), 604
	lock_closed (pymea-
input_grounding (pymeasure.instruments.srs.SR830	sure.instruments.srs.ldc500series.LDC500SeriesLD
property), 577	property), 569
	lock_enabled (pymea-
property), 581	sure.instruments.agilent.AgilentB2981 prop-
input_notch_config (pymea-	erty), 204
	lock_enabled (pymea-
input_range (pymeasure.instruments.srs.SR860 prop-	sure.instruments.novanta.Fpu60 property),
erty), 581	492
•	nal (pymeasure.instruments.hp.hp856Xx.FrequencyReference
• =	A S S S S S S S S S S S S S S S S S S S

attribute), 302	sure.instruments.keithley.Keithley2700
Internal (pymeasure.instruments.hp.hp856Xx.MixerM	
attribute), 301	is_buffer_full() (pymea-
Internal (pymeasure.instruments.hp.hp856Xx.Source	LevelingControd <b>Medu</b> struments.keithley.Keithley6221
attribute), 303	method), 374
internal_frequency (pyme	a- is_buffer_full() (pymea-
sure.instruments.agilent.Agilent8257D pro	p- sure.instruments.keithley.Keithley6517B
erty), 150	method), 380
internal_shape (pyme	a- is_buffer_full() (pymea-
sure.instruments.agilent.Agilent8257D pro	p- sure.instruments.keithley.KeithleyDAQ6510
erty), 150	method), $401$
**	a- is_calibrated (pymea-
sure.instruments.siglenttechnologies.SDS1072	
property), 548	erty), 513
-	a- is_current_stable() (pymea-
sure.instruments.keithley.Keithley2182 pro	p- sure.instruments.danfysik.Danfysik8500
erty), 396	method), 250
<pre>internalfrequency (pymeasure.instruments.srs.SR86</pre>	
property), 582	sure.instruments.ptw.ptwDIAMENTOR prop-
interpolator_autocalibrated (pyme	
	is_enabled(pymeasure.instruments.agilent.Agilent8257D
property), 508	property), 150
	a- is_enabled() (pymea-
	dvantestR624Xsure.instruments.danfysik.Danfysik8500
method), 121	method), 250
<pre>intervall() (pymeasure.instruments.ptw.ptwUNIDC</pre>	= :
method), 514	sure.instruments.keysight.KeysightN5767A
invert (pymeasure.instruments.lecroy.lecroyT3DSO12	
property), 470	is_last() (pymeasure.experiment.procedure.Procedure
invert_signal_sign (pyme	
	is_last() (pymeasure.experiment.workers.Worker
ion_gauge_status (pyme	
	AndPicssmoxGhay()elpymeasure.instruments.parker.ParkerGV6
property), 474	method), 505
	a- is_out_of_range() (pymeasure.instruments.srs.SR830
sure.instruments.mksinst.mks937b), 474	method), 577
	a- is_ready (pymeasure.instruments.siglenttechnologies.SDS1072CML
sure.instruments.oxfordinstruments), 498	property), 548
is_averaging() (pyme	
sure.instruments.agilent.Agilent8722ES	method), 250
method), 152 is_buffer_full() (pyme	is_running() (pymea-
**	a- sure.display.manager.BaseManager method), 79
sure.instruments.keithley.Keithley2000	
method), 326 is_buffer_full() (pyme	
sure.instruments.keithley.Keithley2182	method), 250
method), 396	is_set() (pymeasure.experiment.parameters.Metadata
is_buffer_full() (pyme	
sure.instruments.keithley.Keithley2400	is_set() (pymeasure.experiment.parameters.Parameter
	method), 70
method), 346 is_buffer_full() (pyme	
sure.instruments.keithley.Keithley2450	a- 15_val1u_1esponse() (pymeu- sure.instruments.oxfordinstruments.base.OxfordInstrumentsBase
method), 354	method), 493
is_buffer_full() (pyme	
\$\frac{1}{2} = \frac{1}{2} = \	Pyea

sure.instruments.oxfordinstruments), 494 ITC503.FLOW_CONTROL_STATUS (class in pymea-	Keithley6221 (class in pymea- sure.instruments.keithley), 371
sure.instruments.oxfordinstruments), 494	Keithley6517B (class in pymea- sure.instruments.keithley), 378
J	KeithleyDAQ6510 (class in pymea-
<pre>join() (pymeasure.display.thread.StoppableQThread method), 81</pre>	sure.instruments.keithley), 400 KeithleyDMM6500 (class in pymea-
join() (pymeasure.experiment.workers.Worker method),	sure.instruments.keithley), 383 kelvin (pymeasure.instruments.lakeshore.lakeshore_base.LakeShoreTemp
71	property), 454
K	KepcoBOP3612 (class in pymeasure.instruments.kepco),
Keithley2000 (class in pymea-	407
sure.instruments.keithley), 323	keyboard_locked (pymea-
Keithley2182 (class in pymea- sure.instruments.keithley), 394	sure.instruments.proterial.ROD4 property), 510
Keithley2182Channel (class in pymea-	Keysight81160A (class in pymea-
sure.instruments.keithley.keithley2182), 398	sure.instruments.keysight), 429
Keithley2200 (class in pymea-	Keysight81160A.BaseChannelCreator (class in
sure.instruments.keithley), 330	pymeasure.instruments.keysight), 430
Keithley2200.BaseChannelCreator (class in pymea-	Keysight81160A.ChannelCreator (class in pymea-
sure.instruments.keithley), 330	sure.instruments.keysight), 430 Keysight81160A.MultiChannelCreator (class in
Keithley2200.ChannelCreator (class in pymea-	pymeasure.instruments.keysight), 431
sure.instruments.keithley), 330 Keithley2200.MultiChannelCreator (class in	Keysight81160AChannel (class in pymea-
Keithley2200.MultiChannelCreator (class in pymeasure.instruments.keithley), 331	sure.instruments.keysight.keysight81160A),
Keithley2260B (class in pymea-	439
sure.instruments.keithley), 335	KeysightDSOX1102G (class in pymea-
Keithley2281S (class in pymea-	sure.instruments.keysight), 408
sure.instruments.keithley), 337	KeysightE36312A (class in pymea-
Keithley2281SMeasurementEventRegister	sure.instruments.keysight), 413
(class in pymea-	KeysightE36312A.BaseChannelCreator (class in
sure.instruments.keithley.keithley2281S),	pymeasure.instruments.keysight), 413
340	KeysightE36312A.ChannelCreator (class in pymea- sure.instruments.keysight), 413
Keithley2281SOperationEventRegister	KeysightE36312A.MultiChannelCreator (class in
(class in pymea-	pymeasure.instruments.keysight), 414
sure.instruments.keithley.keithley2281S), 340	KeysightE3631A (class in pymea-
Keithley2306 (class in pymea-	sure.instruments.keysight), 421
sure.instruments.keithley), 341	KeysightE3631A.BaseChannelCreator (class in
Keithley2400 (class in pymea-	pymeasure.instruments.keysight), 421
sure.instruments.keithley), 343	KeysightE3631A.ChannelCreator (class in pymea-
Keithley2450 (class in pymea-	sure.instruments.keysight), 421
sure.instruments.keithley), 352	KeysightE3631A.MultiChannelCreator (class in
Keithley2510 (class in pymea-	pymeasure.instruments.keysight), 422 KeysightN5767A (class in pymea-
sure.instruments.keithley), 359	KeysightN5767A (class in pymea- sure.instruments.keysight), 411
Keithley2600 (class in pymea-	KeysightN7776C (class in pymea-
sure.instruments.keithley), 362	sure.instruments.keysight), 411
Keithley2700 (class in pymea- sure.instruments.keithley), 364	kill() (pymeasure.instruments.parker.ParkerGV6
Keithley2750 (class in pymea-	method), 505
sure.instruments.keithley), 368	kill_enabled (pymea-
Keithley4200 (class in pymea-	sure.instruments.eurotest.EurotestHPP120256
sure.instruments.keithley), 405	property), 255
• •	Kusa245 250A (class in nymea-

sure.instruments.kuhneelectronic), 441	1d (pymeasure.instruments.srs.ldc500series.LDC500Series attribute), 568
L	LD400P (class in pymeasure.instruments.aimtti.ld400p),
labels() (pymeasure.experiment.results.Results	217
method), 72	LDC500Series (class in pymea-
LakeShore211 (class in pymea-	sure.instruments.srs.ldc500series), 568
sure.instruments.lakeshore), 444	LDC500SeriesLD (class in pymea-
LakeShore211.AnalogMode (class in pymea-	sure.instruments.srs.ldc500series), 568
sure.instruments.lakeshore), 444	LDC500SeriesPD (class in pymea-
LakeShore211.AnalogRange (class in pymea-	sure.instruments.srs.ldc500series), 570
sure.instruments.lakeshore), 445	LDC500SeriesTEC (class in pymea-
LakeShore211.RelayMode (class in pymea-	sure.instruments.srs.ldc500series), 570
sure.instruments.lakeshore), 445	LDCCurrent (pymeasure.instruments.thorlabs.ThorlabsPro8000
LakeShore211.RelayNumber (class in pymea-	property), 627
sure.instruments.lakeshore), 445	LDCCurrentLimit (pymea-
LakeShore224 (class in pymea-	sure. instruments. thorlabs. Thorlabs Pro 8000
sure.instruments.lakeshore), 446	property), 627
LakeShore331 (class in pymea-	LDCPolarity (pymeasure.instruments.thorlabs.ThorlabsPro8000
sure.instruments.lakeshore), 448	property), 627
LakeShore3xx (class in pymea-	${\tt LDCStatus} \ (py measure. instruments. thorlabs. Thorlabs Pro 8000$
sure.instruments.lakeshore), 449	property), 627
LakeShore421 (class in pymea-	${\tt learn\_mode} \ (py measure. instruments. temptronic. ATSB as e$
sure.instruments.lakeshore), 451	property), 616
LakeShore425 (class in pymea-	LeCroyT3DS01204 (class in pymea-
sure.instruments.lakeshore), 453	sure.instruments.lecroy), 456
LakeShoreHeaterChannel (class in pymea-	LeCroyT3DSO1204.BaseChannelCreator (class in
sure.instruments.lakeshore.lakeshore_base),	pymeasure.instruments.lecroy), 456
455	LeCroyT3DS01204.ChannelCreator (class in pymea-
LakeShoreTemperatureChannel (class in pymea-	sure.instruments.lecroy), 456
sure.instruments.lakeshore.lakeshore_base),	LeCroyT3DS01204.MultiChannelCreator (class in
454	pymeasure.instruments.lecroy), 457
${\tt lam\_status} \ (py measure. instruments. euro test. Euro test HPR tests of the property of$	Ple6rgyT3DS01204Channel (class in pymea-
property), 255	sure. instruments. lecroy. lecroy T3DSO 1204),
lan_config (pymeasure.instruments.ptw.ptwUNIDOS	470
property), 515	${\tt led}(py measure. instruments. redpitaya. redpitaya\_scpi. RedPitayaScpi$
large_size_enabled (pymea-	attribute), 523
sure.instruments.spellmanhv.spellman XRV. Filam	eheft_limit(pymeasure.instruments.newport.esp300.Axis
property), 567	property), 477
laser_enabled (pymea-	level (pymeasure.instruments.hp.HP8657B property),
sure.instruments.toptica.ibeamsmart.IBeamSmart	
property), 634	level (pymeasure.instruments.rohdeschwarz.sfm.SFM
last_command_error (pymea-	property), 531
sure.instruments.srs.ldc500series.LDC500Series	level_a (pymeasure.instruments.aimtti.ld400p.LD400P
property), 568	property), 217
last_execution_error (pymea-	level_b (pymeasure.instruments.aimtti.ld400p.LD400P
sure.instruments.srs.ldc500series.LDC500Series	property), 218
property), 568	level_lin(pymeasure.instruments.anritsu.AnritsuMS9710C
last_test_date (pymea-	property), 231
sure.instruments.tdk.tdk_gen40_38.TDK_Gen40_	3&vel_log(pymeasure.instruments.anritsu.AnritsuMS9710C
property), 591	property), 231
last_test_date (pymea-	level_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM
sure.instruments.tdk.tdk_gen80_65.TDK_Gen80_	65 property), 531
property), 595	level_offset (pymeasure.instruments.hp.HP8657B property), 306

level_opt_attn (pymea- sure.instruments.anritsu.AnritsuMS9710C	attribute), 196
property), 231	linear_display_enabled (pymea-
level_position (pymea-	
sure.instruments.yokogawa.aq6370series.AQ63	770 <b>Sanea</b> r_Double (pymea-
property), 641	sure.instruments.agilent.agilentB1500.SweepMode
<pre>level_scale(pymeasure.instruments.anritsu.AnritsuMS</pre>	S9710C attribute), 196
property), 231	LINEAR_SINGLE (pymea-
level_select (pymea-	
sure.instruments.aimtti.ld400p.LD400P	attribute), 196
property), 218	LineEditDelegate (class in pymea-
lia_status (pymeasure.instruments.srs.SR830 prop-	• • • • • • • • • • • • • • • • • • • •
erty), 577	85
limit (pymeasure.instruments.spellmanhv.spellmanXRV.	Fillingutfiles() (pymea-
property), 567	sure.instruments.activetechnologies.AWG401x_AWG
limit_enabled (pymeasure.instruments.hp.HP8116A	
property), 272	list_resources() (in module pymeasure.instruments),
limit_high (pymeasure.instruments.hp.HP437B prop-	
erty), 312	Listener (class in pymeasure.experiment.listeners), 64
limit_high (pymeasure.instruments.keysight.keysight8)	
property), 440	$I_{\mathcal{I}}$
limit_high_hit (pymeasure.instruments.hp.HP437E	
property), 312	lo_frequency (pymea-
limit_low (pymeasure.instruments.hp.HP437B prop-	
erty), 312	tribute), 292
	60AcKet (spight8dsli60eAiCstaumednts.rigol.rigol_dg800.VoltageChannel
property), 441	property), 525
limit_low_hit (pymeasure.instruments.hp.HP437E property), 312	method), 79
limit_state_enabled (pymea-	
sure.instruments.keysight.keysight81160A.Keys	·
property), 441	<pre>load() (pymeasure.display.widgets.image_widget.ImageWidget</pre>
limits_enabled (pymeasure.instruments.hp.HP437E	
property), 312	<pre>load() (pymeasure.display.widgets.plot_widget.PlotWidget</pre>
Line (pymeasure.instruments.hp.hp856Xx.TriggerMode	
attribute), 304	<pre>load() (pymeasure.display.widgets.tab_widget.TabWidget</pre>
line_frequency (pymea-	method), 88
sure.instruments.advantest.advantestR624X.AdvantestR64X.AdvantestR64X.AdvantestR64X.AdvantestR64X.AdvantestR6	van <b>keaR624X</b> ymeasure.display.widgets.table_widget.TableWidget
property), 126	method), 90
line_frequency (pymea-	- load() (pymeasure.experiment.results.Results static
sure.instruments.keithley.Keithley2182 prop-	- <i>method</i> ), 72
erty), 396	load_capacity (pymea-
line_frequency (pymea-	sure.instruments.tcpowerconversion.CXN
sure.instruments.keithley.Keithley2281S	property), 586
property), 339	load_capacity (pymea-
line_frequency (pymea-	
sure.instruments.keithley.Keithley2400 prop-	
erty), 346	load_config(pymeasure.instruments.advantest.advantestR624X.Advan
line_frequency (pymea-	
sure.instruments.keithley.KeithleyDMM6500	load_data_file() (pymea-
property), 388	sure.instruments.anritsu.AnritsuMS464xB
line_frequency_auto (pymea-	
sure.instruments.keithley.Keithley2400 prop-	
erty), 346	sure.instruments.anritsu.AnritsuMS464xB
- : ///	

method), 240		${\tt log\_ratio} \ (py measure. instruments. signal recovery. DSP7265$
load_enabled	(рутеа-	property), 561
	Agilent428	8 <b>4.0Go&amp;AMGoE</b> (pymeasure.instruments.agilent.agilentB1500.SweepMode
property), 200		attribute), 196
load_function		logarithmic_scale (pymea-
<u> </u>	Agilent428	84ACorrect <b>san</b> e.instruments.hp.hp856Xx.HP856Xx at-
property), 200		tribute), 276
load_function		LogHandler (class in pymeasure.display.log), 79
sure.instruments.agilent.agilent4284A. property), 200	Agilent428	8 <b>4.4.5Han</b> dler.Emitter (class in pymeasure.display.log),
load_impedance	(nymaa	LogWidget (class in pymea-
		Logwidget (class in pymea- hannelAFGsure.display.widgets.log_widget), 84
property), 115	10101x.CI	low_freq (pymeasure.instruments.srs.SR570 property),
load_sequence()	(рутеа-	574
sure.display.widgets.sequencer_widget		
method), 87	.sequence	sure.instruments.agilent.Agilent8257D prop-
load_sequence()	(рутеа-	erty), 150
sure.instruments.rohdeschwarz.hmp.H		low_freq_out_source (pymea-
method), 543	WII 4040	sure.instruments.agilent.Agilent8257D prop-
load_setup() (pymeasure.instruments.s	rc SR830	erty), 151
method), 577	73.5K050	low_level (pymeasure.instruments.hp.HP8116A prop-
load_setup_file	(рутеа-	erty), 272
sure.instruments.temptronic.ATSBase		lower_sideband_enabled (pymea-
erty), 616	prop-	sure.instruments.rohdeschwarz.sfm.SFM
local() (pymeasure.instruments.aimtti.aimttiPi	I DI Rasa	
method), 215	L.I LDuse	property), 531 lvps_monitor (pymea-
local() (pymeasure.instruments.danfysik.Danj	Sveil-8500	lvps_monitor (pymea- sure.instruments.spellmanhv.spellmanXRV.UnscaledData
method), 250	ysikosoo	
	hlav2000	property), 567
local() (pymeasure.instruments.keithley.Keit method), 326	niey2000	M
local_lockout	(рутеа-	
sure.instruments.temptronic.ATSBase	4.7	mac_address (pymeasure.instruments.ptw.ptwUNIDOS
erty), 617	prop-	property), 515
locked (pymeasure.instruments.keysight.Keysig.	h+N7776C	mag (pymeasure.instruments.ametek.Ametek7270 prop-
property), 412	mini	erty), 221
LOG_10 (pymeasure.instruments.agilent.agilentE	21500 Sam	mag (pymeasure.instruments.signalrecovery.DSP7225
attribute) 106	01300.Sam <sub>i</sub>	
attribute), 196 LOG_100 (pymeasure.instruments.agilent.agilent	D1500 Car	mag (pymeasure.instruments.signalrecovery.DSP7265
attribute), 196	B1300.Sar	
LOG_25 (pymeasure.instruments.agilent.agilentE	21500 Sam	magnet_current (pymeasure.instruments.ami.AMI430
	1300.Sam	1 1 27
attribute), 196	D1500 Car	magnet_status (pymeasure.instruments.ami.AMI430
LOG_250 (pymeasure.instruments.agilent.agilent	B1300.Sar	1 1 1 · · ·
attribute), 196	0.1500 C	MagnetError (class in pymea-
LOG_50 (pymeasure.instruments.agttent.agttent.b	1300.Sam	plingMode sure.instruments.oxfordinstruments.ips120_10),
attribute), 196		501
LOG_5000 (pymeasure.instruments.agilent.agilei	ntB1500.Sc	malins Unde (pymeasure.instruments.srs.SR830 property),
attribute), 196	·1 .D1500	577
LOG_DOUBLE (pymeasure.instruments.aguent.agu	<i>uentB1300</i>	Sixten Made (pymeasure.instruments.srs.SR860 property),
attribute), 196	(	582
log_magnitude()	(pymea-	magnitude() (pymeasure.instruments.agilent.Agilent8722ES
sure.instruments.agilent.Agilent8722E	S	method), 152
method), 152	DCD73	main_air_flow_rate (pymea-
log_ratio (pymeasure.instruments.signalrecov	ery.DSP/2	sure.msn unemsnemproneal 1 sbase prop
property), 554		erty), 617

ManagedConsole (class in pymeasure.display 75	.console),	sure.instruments.tdk.tdk_ property), 591	_gen40_38.TDK_Gen40_38
ManagedDockWindow (class in	рутеа-	master_slave_setting	(pymea-
sure.display.windows.managed_dock_ 94	window),	sure.instruments.tdk.tdk_ property), 595	_gen80_65.TDK_Gen80_65
ManagedImageWindow (class in	рутеа-	material (pymeasure.instrument	ts.aja.DCXS property),
sure.display.windows.managed_image	_window),	219	
91		math_define(pymeasure.instrum	nents.lecroy.LeCroyT3DSO1204
ManagedWindow (class in	рутеа-	property), 462	
sure.display.windows.managed_windo 91	ow),	math_mode (pymeasure.instru property), 519	ments.racal.Racal1992
ManagedWindowBase (class in	рутеа-	math_vdiv(pymeasure.instrumen	ts.lecroy.LeCroyT3DSO1204
sure.display.windows.managed_windo	ow),	property), 463	
92		math_vpos( <i>pymeasure.instrumen</i>	ts.lecroy.LeCroyT3DSO1204
Manager (class in pymeasure.display.manager)		property), 463	
manu (pymeasure.instruments.hp.HP8753E p 308	property),	math_x (pymeasure.instruments.i erty), 519	racal.Racal1992 prop-
MANUAL (pymeasure.instruments.agilent.agilenti attribute), 195	B1500.ADC	<b>Math_z</b> (pymeasure.instruments.i erty), 519	racal.Racal1992 prop-
MANUAL (pymeasure.instruments.agilent.agilenti attribute), 195	B1500.Auto	Maxualmplitude (pymeasure.inst property), 262	truments.hp.HP33120A
MANUAL (pymeasure.instruments.agilent.agilent)	B1500.Com	nlancentoku ingount	(pymea-
attribute), 197		sure.instruments.hp.HP3	33120A property),
MANUAL (pymeasure.instruments.hp.hp856Xx.An attribute), 300	nplitudeUn	ts 262 max_burst_phase	(pymea-
manual_mode (pymeasure.instruments.tcpowere	conversion.	_	~ -
property), 587		262	F F 1, 7,
manual_trigger_type	(рутеа-	max_burst_rate	(pymea-
sure.instruments.anritsu.AnritsuMS46	64xB	sure.instruments.hp.HP3	33120A property),
property), 240		262	
manufacturer	(pymea-	max_current(pymeasure.instrum	nents.deltaelektronika.SM7045D
sure.instruments.mksinst.mks974b.MK	KS974B	property), 252	
property), 475		max_current(pymeasure.instrum	ents.keithley.Keithley2400
marker_amplitude	(pymea-	property), 346	
sure.instruments.hp.hp856Xx.HP856X tribute), 289	Xx at-	max_current (pymeasure.instrum property), 354	nents.keithley.Keithley2450
marker_delta		max_current(pymeasure.instrum	nents.rohdeschwarz.hmp.HMP4040
sure.instruments.hp.hp856Xx.HP856X	ζx at-	property), 543	
tribute), 289		max_frequency (pymeasure.inst	truments.hp.HP33120A
marker_frequency	(pymea-	property), 263	
sure.instruments.hp.hp856Xx.HP856X	Xx at-	max_hold_enabled	(pymea-
tribute), 290		sure.instruments.lakesho	ore.LakeShore421
marker_noise_mode_enabled	(рутеа-	property), 452	
sure.instruments.hp.hp856Xx.HP856X	Xx at-	max_hold_field	(pymea-
tribute), 290		sure.instruments.lakesho	ore.LakeShore421
marker_signal_tracking_enabled	(рутеа-	property), 452	
sure.instruments.hp.hp856Xx.HP856X	Xx at-	max_hold_field_raw	(pymea-
tribute), 291		sure.instruments.lakesho	ore.LakeShore421
marker_threshold	(pymea-	property), 452	,
sure.instruments.hp.hp856Xx.HP856X	Xx at-	max_hold_multiplier	(pymea-
tribute), 291	/W ****	sure.instruments.lakesho	ore.LakeShore421
marker_time(pymeasure.instruments.hp.hp850	6 <i>Xx.HP</i> 856		,
attribute), 291	,	<pre>max_hold_reset()</pre>	(pymea-
master_slave_setting	(pymea-	sure.instruments.lakesho	re.LakeShore421

method), 452			sure.instruments.keithley.Keithley2400	prop-
max_number_of_points	(pymea-		erty), 346	
sure.instruments.anritsu.AnritsuMS46	4xB	mean_vo	ltage	(pymea-
property), 240			sure. instruments. keithley. Keithley 2450	prop-
<pre>max_offset (pymeasure.instruments.hp.H</pre>	P33120A		erty), 355	
property), 263		means	(pymeasure.instruments.keithley.Keith	aley2400
<pre>max_output_amplitude</pre>	(pymea-		property), 346	
sure.instruments.teledyne.teledyneT3A	FG.Signal	Cheans l	(pymeasure.instruments.keithley.Keith	aley2450
property), 602			property), 355	/
max_power (pymeasure.instruments.oxfordinstru	uments.mer	CME ASC. AC		(pymea-
property), 503	(		sure.instruments.anritsu.AnritsuMS209	0A
max_resistance	(pymea-		property), 235	(
sure.instruments.keithley.Keithley2400	prop-	meas_em		(pymea-
erty), 346	(		sure.instruments.anritsu.AnritsuMS209	UA
max_resistance	(pymea-		property), 235	(mayor a a
sure.instruments.keithley.Keithley2450	prop-	meas_em	sure.instruments.anritsu.AnritsuMS209	(pymea- 04
erty), 354 max_voltage (pymeasure.instruments.deltaelek	tronika SM	17045D	property), 236	UA .
property), 252	aronika.Siv			(рутеа-
max_voltage(pymeasure.instruments.keithley.l	Koithlov240		sure.instruments.anritsu.AnritsuMS209	
property), 346	Xeiiiie y 2+0	<i>,</i>	property), 236	071
max_voltage (pymeasure.instruments.keithley.l	Keithlev24	Smeas in		(рутеа-
property), 354	Actinic y 2+3	wicus_III	sure.instruments.anritsu.AnritsuMS209	
max_voltage (pymeasure.instruments.rohdesch	warz hmp	HMP4040		071
property), 543	man zimipi.			(рутеа-
maximum (pymeasure.instruments.keithley.Keit	hlev2182		sure.instruments.anritsu.AnritsuMS209	
property), 396			property), 236	
maximum_case_temperature	(рутеа-	meas_iq		(рутеа-
sure.instruments.ipgphotonics.yar.YAF		•	sure.instruments.anritsu.AnritsuMS209	
erty), 322			property), 236	
maximum_test_time	(рутеа-	meas_mo	de() (pymeasure.instruments.agilent.ag	ilentB1500.AgilentB1500
sure.instruments.temptronic.ATSB ase	prop-		method), 186	
erty), 617		meas_op	_mode	(рутеа-
${\tt maximums}\ (pymeasure. instruments. keithley. Keithlight and the property of the property$	hley2400		sure.instruments.agilent.agilentB1500.5	SMU
property), 346			property), 190	
${\tt maximums}\ (pymeasure. instruments. keithley. Keit$	hley2450	meas_ot		(рутеа-
property), 354			sure.instruments.anritsu.AnritsuMS209	OA
$\verb maxspeed  (pymeasure. instruments. an a heimauto)   $	mation.DF	PSeriesMoi	tqn <b>6patt</b> y][ <b>2</b> 36	
property), 225		meas_ot		(pymea-
mean (pymeasure.instruments.keithley.Keit	hley2182		sure.instruments.anritsu.AnritsuMS209	OA
property), 396			property), 236	
mean_current	(pymea-	meas_po	wer (pymeasure.instruments.anritsu.Anr	ritsuMS2090A
sure.instruments.keithley.Keithley2400	) prop-		property), 236	,
erty), 346	,	meas_po		(pymea-
mean_current	(pymea-		sure.instruments.anritsu.AnritsuMS209	OA
sure.instruments.keithley.Keithley2450	) prop-		property), 236	/
erty), 354	(	meas_ra	5	(pymea-
mean_resistance	(pymea-		sure.instruments.agilent.agilentB1500.5	DIVI U
sure.instruments.keithley.Keithley2400	) prop-	moss rs	property), 191	(mymag
<pre>erty), 346 mean_resistance</pre>	(рутеа-	meas_ra		(pymea- SMI)
sure.instruments.keithley.Keithley2450			sure.instruments.agilent.agilentB1500.2 method), 191	JIVI U
erty), 354	, ргор-	meas ra	_	(рутеа-
mean_voltage	(рутеа-	™cu∋_1 a	sure.instruments.agilent.agilentB1500.S	* *
	47		5	

	property), 191			method),	402		
MeasMode		рутеа-				(рутеа-	
	sure.instruments.agilent.agilentB150				uments.keithley.Kei	* *	
MeasOpMo		рутеа-		method),		•	
_	sure.instruments.agilent.agilentB150				(pymeasure.instrur	nents.hp.HP3478A	
Measural		рутеа-		property)		1	
	sure.experiment.parameters), 69		measure_		(pymeasure.instrur	nents.hp.HP3478A	
	(pymeasure.instruments.agilent.Ag			property)		1	
	property), 204		measure_			(pymea-	
	() (pymeasure.instruments.agilent.ag	gilent4156.A		-	ruments.lecroy.LeCr	royT3DSO1204	
	method), 168			property)			
measure	() (pymeasure.instruments.lakeshore	.LakeShore4	2maeasure	_diode()	)	(pymea-	
	method), 454			sure.instr	uments.keithley.Kei	ithley2000	
measure	() (pymeasure.instruments.ptw.p.	tw <i>UNIDOS</i>		method),	326		
	method), 515		measure.	_diode()	1	(pymea-	
measure_	_ACI (pymeasure.instruments.hp	o.HP3478A		sure.instr	uments.keithley.Kei	ithleyDMM6500	
	property), 269			method),	388		
measure_	_ACV (pymeasure.instruments.hp	o.HP3478A	measure_	_frequen	ıcy()	(pymea-	
	property), 269			sure.instr	uments.keithley.Kei	ithley2000	
measure_	_capacitance()	(pymea-		method),	326		
	sure.instruments.keithley.Keithley Dlaws and the property of	MM6500	measure_	_frequen	ıcy()	(pymea-	
	method), 388			sure.instr	uments.keithley.Kei	ithleyDMM6500	
measure_	_capacity()	(pymea-		method),	388		
	sure.instruments.attocube.anc300.A.	xis	measure_	_load()		(pymea-	
	method), 247			sure.instr	ruments.agilent.agil	ent4284A.Agilent4284ACorrection	ţ
measure	_concurent_functions	(pymea-		method),	200		
	sure. instruments. keithley. Keithley 24	00 prop-	measure.	_load()		(pymea-	
	erty), 347			sure.instr	uments.agilent.agil	ent4284A.Agilent4284ASpot	
	_continuity()	(pymea-		method),	200		
	sure.instruments.keithley.Keithley20	00	measure_			(pymea-	
	method), 326				uments.anritsu.Anr	itsuMS9710C	
	_continuity()	(pymea-		property)	, 231		
	sure. instruments. keithley. Keithley Dlaws and the property of the property	MM6500				(рутеа-	
	method), 388			sure.instr	uments.agilent.agil	ent4284A.Agilent4284ACorrection	,
	_current	(pymea-		method),	200		
	sure. instruments. delta elektronika. SM	17045D		_		(pymea-	
	property), 252					ent4284A.Agilent4284ASpot	
	_current()	(рутеа-		method),			
	sure.instruments.advantest.advantes	tR624X.SMU		_		(рутеа-	
	method), 135				ruments.lecroy.LeC1	oyT3DSO1204	
	_current()	(pymea-		method),			
	sure.instruments.keithley.Keithley20	00	measure_			(рутеа-	
	method), 326	,			•	edyne_oscilloscope.TeledyneOscill	os
	_current()	(pymea-		method),			
	sure.instruments.keithley.Keithley24	00	measure_			(pymea-	
	method), 347	,			ruments.teledyne.Tel	ledyneOscilloscope	
	_current()	(pymea-		method),	604		
	sure.instruments.keithley.Keithley24	50	measure_	_		(pymea-	
	method), 355	(			uments.anritsu.Anr	usuMS9/10C	
	_current()	(pymea-		method),			
	sure.instruments.keithley.Keithley65	1/ <b>B</b>	measure_	_		(pymea-	
	method), 380	(			ruments.keithley.Kei	iniey2000	
	_current()	(pymea-		method),			
	sure.instruments.keithley.KeithleyDA	IQ6510	measure_	_period(	.)	(рутеа-	

	sure.instruments.keithley.KeithleyDMl method), 388	M6500	sure.instruments.keithley.Keithley65. method), 380	17B
measure	_R2W (pymeasure.instruments.hp.l	HP3478A	<pre>measure_voltage()</pre>	(рутеа-
	property), 270		sure.instruments.keithley.KeithleyDA	Q6510
measure		HP3478A	method), 402	~
	property), 270		<pre>measure_voltage()</pre>	(рутеа-
measure	_resistance()	(рутеа-	sure.instruments.keithley.KeithleyDM	* *
	sure.instruments.keithley.Keithley2000		method), 389	
	method), 326		measured_current	(рутеа-
measure	_resistance()	(рутеа-	sure.instruments.rohdeschwarz.hmp.	* *
meabar e	sure.instruments.keithley.Keithley2400		property), 543	11111 1010
	method), 347	,	measured_value	(рутеа-
masciira	_resistance()	(рутеа-	sure.instruments.racal.Racal1992	property),
illeasur e			520	property),
	sure.instruments.keithley.Keithley2450	,		(
	method), 355	(	measured_voltage	(pymea-
measure	_resistance()	(pymea-	sure.instruments.rohdeschwarz.hmp.	HMP4040
	sure.instruments.keithley.Keithley6517	'B	property), 543	DILLIGHTOR
	method), 380	,	measurement (pymeasure.instruments.ptw.ptw	DIAMENTOR
measure	_resistance()	(рутеа-	property), 513	
	sure.instruments.keithley.KeithleyDAQ	96510	measurement()	(рутеа-
	method), $402$		sure.instruments.common_base.Com	monBase
measure	_resistance()	(рутеа-	static method), 101	
	sure.instruments.keithley.KeithleyDM	M6500	<pre>measurement()</pre>	(pymea-
	method), 388		sure.instruments.keysight.Keysight81	! 160A
measure	_Rext (pymeasure.instruments.hp.)	HP3478A	static method), 436	
	property), 270		<pre>measurement()</pre>	(рутеа-
measure	_short()	(рутеа-	sure.instruments.keysight.KeysightE	36312A
	sure.instruments.agilent.agilent4284A	.Agilent428	84ACorrectistatic method), 418	
	method), 200		measurement()	(pymea-
measure	_short()	(рутеа-	sure.instruments.keysight.KeysightE	3631A
	sure.instruments.agilent.agilent4284A			
	method), 201		measurement()	(рутеа-
measure	_temperature()	(рутеа-	sure.instruments.lecroy.LeCroyT3DS	* *
cubur c	sure.instruments.keithley.Keithley2000		static method), 463	.01207
	method), 326	,	measurement_count	(рутеа-
masciira	_temperature()	(рутеа-	sure.instruments.advantest.advantest	
measure	sure.instruments.keithley.KeithleyDMl		property), 137	ROZTA.SMO Channel
		M0300		(mym a g
	method), 389	(	measurement_event	(pymea-
measure	_voltage	(pymea-	sure.instruments.keithley.Keithley226	818
	sure.instruments.deltaelektronika.SM7	1043D	property), 339	
	property), 252	,	measurement_event_enabled	(pymea-
measure	_voltage()	(рутеа-	sure.instruments.keithley.Keithley622	21 prop-
	sure.instruments.advantest.advantestR	624X.SMU	• **	
	method), 130		measurement_events	(рутеа-
measure	_voltage()	(рутеа-	sure.instruments.keithley.Keithley62	21 prop-
	sure.instruments.keithley.Keithley2000	)	erty), 374	
	method), 326		measurement_history	(pymea-
measure	_voltage()	(рутеа-	sure.instruments.ptw.ptwUNIDOS	property),
	sure.instruments.keithley.Keithley2400	)	515	
	method), 347		measurement_ongoing	(pymea-
measure	_voltage()	(рутеа-	sure.instruments.keithley.Keithley22	
	sure.instruments.keithley.Keithley2450		property), 339	
	method), 355		measurement_parameter	(рутеа-
measure	_voltage()	(pymea-	sure.instruments.anritsu.anritsuMS4	

property), 245		sure.experiment.procedure.Proced	lure method),	
measurement_parameters	(рутеа-	65		
sure.instruments.ptw.ptwUNIDOS 515	property),	mfc_range (pymeasure.instruments.proterial.rod4.ROD4Channe property), 511		
measurement_result	(рутеа-	min_amplitude (pymeasure.instruments.h	р.НР33120А	
	property),	property), 263	•	
515		min_burst_count	(pymea-	
measurement_settings	(рутеа-	sure.instruments.hp.HP33120A	property),	
sure.instruments.philips.PM6669	property),	263		
507		min_burst_phase	(pymea-	
measurement_time	(рутеа-	sure.instruments.hp.HP33120A	property),	
sure.instruments.pendulum.cnt91.CN	Т91	263		
property), 508		min_burst_rate	(рутеа-	
measurement_time	(рутеа-	sure.instruments.hp.HP33120A	property),	
sure.instruments.philips.PM6669	property),	263		
507		min_current (pymeasure.instruments.keith	ley.Keithley2400	
measurement_timeout	(рутеа-	property), 347		
sure.instruments.philips.PM6669 507	property),	min_current (pymeasure.instruments.keith property), 355	ley.Keithley2450	
measurement_type	(рутеа-	<pre>min_current(pymeasure.instruments.rohd</pre>	eschwarz.hmp.HMP4040	
sure.instruments.agilent.Agilent4294A	A prop-	property), 543		
erty), 172		min_frequency (pymeasure.instruments.h	p.HP33120A	
measurement_unit	(рутеа-	property), 263		
sure.instruments.hp.HP437B property	v), 313	min_offset (pymeasure.instruments.h	p.HP33120A	
MeasurementChannel (class in	рутеа-	property), 263		
sure.instruments.anritsu.anritsuMS46	(4xB),	min_resistance	(pymea-	
242		sure.instruments.keithley.Keithley.	2400 prop-	
MeasurementType (class in	рутеа-	erty), 347		
sure.instruments.advantest.advantest Properties (a) and the state of	R624X),	min_resistance	(рутеа-	
140		sure.instruments.keithley.Keithley2	2450 prop-	
measuring_function	(рутеа-	erty), 355		
	property),	<pre>min_voltage(pymeasure.instruments.keith</pre>	ley.Keithley2400	
507		property), 347		
measuring_parameter	(рутеа-	min_voltage(pymeasure.instruments.keith	ley.Keithley2450	
	property),	property), 355		
308		min_voltage(pymeasure.instruments.rohd	eschwarz.hmp.HMP4040	
$\verb memory_size   (pymeasure.instruments.lecroy.L$	eCroyT3DS	* * * ·		
property), 464		minimum (pymeasure.instruments.keithley.	Keithley2182	
$\verb memory_size   (pymeasure.instruments.teledynes) $	.TeledyneO			
property), 604		minimum_display_power	(рутеа-	
menu (pymeasure.instruments.lecroy.LeCroyT3	DSO1204	sure.instruments.ipgphotonics.yar.	YAR prop-	
property), 464		erty), 322		
MercuryiTC (class in	рутеа-	minimums (pymeasure.instruments.keithley.	Keithley2400	
sure.instruments.oxfordinstruments),				
MESSACE (nymageura instruments hn hn856Vr)		property), 347		
attribute), 303		teminimums (pymeasure.instruments.keithley. property), 355		
		teminimums (pymeasure.instruments.keithley.		
attribute), 303	StatusRegisi	teminimums (pymeasure.instruments.keithley. property), 355		
<pre>attribute), 303 message_waiting()</pre>	StatusRegist (pymea-	teminimums (pymeasure.instruments.keithley. property), 355 mixer_bias (pymeasure.instruments.hp.HF		
attribute), 303 message_waiting() sure.experiment.listeners.Listener	StatusRegiss (pymea- method),	teminimums (pymeasure.instruments.keithley. property), 355 mixer_bias (pymeasure.instruments.hp.HF erty), 299	P8561B prop-	
attribute), 303 message_waiting() sure.experiment.listeners.Listener 64 Metadata (class in pymeasure.experiment.pa	(pymea- method), rameters),	teminimums (pymeasure.instruments.keithley. property), 355 mixer_bias (pymeasure.instruments.hp.HF erty), 299 mixer_bias_enabled sure.instruments.hp.HP8561B	P8561B prop- (pymea- property),	

```
erty), 299
                                                            property), 537
MixerMode
                  (class
                                                   modulation_enabled
                                                                                              (рутеа-
                                in
                                           рутеа-
        sure.instruments.hp.hp856Xx), 301
                                                            sure.instruments.srs.ldc500series.LDC500SeriesLD
MKS937B
                 (class
                                                            property), 569
                                in
                                           рутеа-
        sure.instruments.mksinst.mks937b), 472
                                                   module
MKS974B
                                                       pymeasure.display.browser, 75
                 (class
                                in
                                           рутеа-
        sure.instruments.mksinst.mks974b), 474
                                                       pymeasure.display.console, 75
                                                       pymeasure.display.curves, 76
MKSInstrument
                      (class
                                           рутеа-
        sure.instruments.mksinst.mksinst), 471
                                                       pymeasure.display.inputs, 77
                                                       pymeasure.display.listeners, 79
mode
        (pymeasure.instruments.agilent.Agilent34450A
        property), 164
                                                       pymeasure.display.log, 79
mode (pymeasure.instruments.agilent.AgilentE4980 prop-
                                                       pymeasure.display.manager, 79
        erty), 154
                                                       pymeasure.display.plotter, 81
        (pymeasure.instruments.aimtti.ld400p.LD400P
                                                       pymeasure.display.thread, 81
mode
                                                       pymeasure.display.widgets.browser_widget,
        property), 218
mode
         (pymeasure.instruments.attocube.anc300.Axis
                                                       pymeasure.display.widgets.directory_widget,
        property), 247
      (pymeasure.instruments.hp.HP3478A property),
mode
                                                       pymeasure.display.widgets.dock_widget, 88
mode
         (pymeasure.instruments.keithley.Keithley2000
                                                       pymeasure.display.widgets.estimator_widget,
        property), 326
mode (pymeasure.instruments.keithley.KeithleyDMM6500
                                                       pymeasure.display.widgets.fileinput_widget,
        property), 389
mode (pymeasure.instruments.keithley.keithleyDMM6500.ScannepQmed2000Cldiusp2lay.widgets.filename_widget,
        property), 394
mode
       (pymeasure.instruments.lakeshore.LakeShore425
                                                       pymeasure.display.widgets.image_frame, 83
        property), 454
                                                       pymeasure.display.widgets.image_widget,
mode (pymeasure.instruments.srs.ldc500series.LDC500SeriesLD
        property), 569
                                                       pymeasure.display.widgets.inputs_widget,
mode (pymeasure.instruments.srs.ldc500series.LDC500SeriesTEC
        property), 571
                                                       pymeasure.display.widgets.log_widget, 84
mode (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38pymeasure.display.widgets.plot_frame, 84
                                                       pymeasure.display.widgets.plot_widget, 84
        property), 591
mode(pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65pymeasure.display.widgets.results_dialog,
        property), 595
mode (pymeasure.instruments.temptronic.ATSBase prop-
                                                       pymeasure.display.widgets.sequencer_widget,
mode\ (pymeasure.instruments.thermotron.Thermotron3800
                                                       pymeasure.display.widgets.tab_widget,87
        property), 625
                                                       pymeasure.display.widgets.table_widget,
model (pymeasure.instruments.hp.HP8753E property),
                                                       pymeasure.display.windows.managed_dock_window,
model (pymeasure.instruments.mksinst.mks974b.MKS974B
                                                       pymeasure.display.windows.managed_image_window,
        property), 475
modulation_bandwidth
                                          (pymea-
        sure.instruments.srs.ldc500series.LDC500SeriesLD
                                                       pymeasure.display.windows.managed_window,
        property), 569
                                                       pymeasure.display.windows.plotter_window,
modulation_degree
                                          (pymea-
        sure.instruments.rohdeschwarz.sfm.Sound_Channel
        property), 537
                                                       pymeasure.experiment.experiment, 63
modulation_enabled
                                          (рутеа-
                                                       pymeasure.experiment.listeners, 64
        sure.instruments.rohdeschwarz.sfm.SFM
                                                       pymeasure.experiment.parameters, 67
        property), 531
                                                       pymeasure.experiment.procedure, 65
modulation_enabled
                                          (рутеа-
                                                       pymeasure.experiment.results, 72
        sure.instruments.rohdeschwarz.sfm.Sound Channel
                                                       pymeasure.experiment.workers, 71
```

```
pymeasure.instruments, 95
                                                 pymeasure.instruments.philips, 506
pymeasure.instruments.activetechnologies,
                                                 pymeasure.instruments.proterial, 508
                                                 pymeasure.instruments.ptw, 511
pymeasure.instruments.aculight, 116
                                                 pymeasure.instruments.racal, 518
pymeasure.instruments.advantest, 117
                                                 pymeasure.instruments.razorbill, 521
pymeasure.instruments.advantest.advantestR37676@measure.instruments.redpitaya, 522
                                                 pymeasure.instruments.rigol, 524
pymeasure.instruments.advantest.advantestR624Xpymeasure.instruments.rohdeschwarz, 525
    139
                                                 pymeasure.instruments.santec, 544
pymeasure.instruments.agilent, 148
                                                 pymeasure.instruments.siglenttechnologies,
pymeasure.instruments.agilent.agilent4156,
                                                 pymeasure.instruments.signalrecovery, 550
pymeasure.instruments.agilent.agilentB1500,
                                                 pymeasure.instruments.spellmanhv, 564
    194
                                                 pymeasure.instruments.srs, 567
pymeasure.instruments.aimtti, 215
                                                 pymeasure.instruments.tcpowerconversion,
pymeasure.instruments.aja, 218
                                                     585
pymeasure.instruments.ametek, 220
                                                 pymeasure.instruments.tdk, 589
pymeasure.instruments.ami, 222
                                                 pymeasure.instruments.tektronix, 598
pymeasure.instruments.anaheimautomation,
                                                 pymeasure.instruments.teledyne, 599
                                                 pymeasure.instruments.temptronic, 613
pymeasure.instruments.anapico, 226
                                                 pymeasure.instruments.texio, 622
pymeasure.instruments.andeenhagerling,
                                                 pymeasure.instruments.thermotron, 625
    227
                                                 pymeasure.instruments.thorlabs, 626
pymeasure.instruments.anritsu, 230
                                                 pymeasure.instruments.thvracont.627
pymeasure.instruments.attocube, 245
                                                 pymeasure.instruments.toptica, 633
pymeasure.instruments.bkprecision, 248
                                                 pymeasure.instruments.validators, 108
pymeasure.instruments.comedi, 110
                                                 pymeasure.instruments.velleman, 635
pymeasure.instruments.danfysik, 249
                                                 pymeasure.instruments.yokogawa, 637
pymeasure.instruments.deltaelektronika,
                                                 pymeasure.test, 57
    252
                                             Monitor (class in pymeasure.display.listeners), 79
pymeasure.instruments.edwards, 253
                                             Monitor (class in pymeasure.experiment.listeners), 64
pymeasure.instruments.eurotest, 253
                                             monitored_value (pymeasure.instruments.hcp.TC038
pymeasure.instruments.fluke, 256
                                                     property), 260
pymeasure.instruments.fwbell, 256
                                             motion_done(pymeasure.instruments.newport.esp300.Axis
pymeasure.instruments.hcp, 260
                                                     property), 477
pymeasure.instruments.heidenhain, 259
                                             mouseMoved()
                                                            (pymeasure.display.curves.Crosshairs
pymeasure.instruments.hp, 261
                                                     method), 76
pymeasure.instruments.inficon, 319
                                             mout (pymeasure.instruments.lakeshore.lakeshore_base.LakeShoreHeaterC
pymeasure.instruments.ipgphotonics, 321
                                                     property), 455
pymeasure.instruments.keithley, 322
                                             move() (pymeasure.instruments.anaheimautomation.DPSeriesMotorContro
pymeasure.instruments.kepco, 407
                                                     method), 226
pymeasure.instruments.keysight, 408
                                             move()
                                                      (pymeasure.instruments.attocube.anc300.Axis
pymeasure.instruments.kuhneelectronic,
                                                     method), 247
    441
                                                         (pymeasure.instruments.parker.ParkerGV6
                                             move()
pymeasure.instruments.lakeshore, 444
                                                     method), 505
pymeasure.instruments.lecroy, 455
                                             move_raw() (pymeasure.instruments.attocube.anc300.Axis
pymeasure.instruments.mksinst, 470
                                                     method), 247
pymeasure.instruments.newport, 476
                                             mroll_frequency
                                                                                     (pymea-
pymeasure.instruments.ni, 478
                                                     sure.instruments.hp.hp856Xx.HP856Xx
                                                                                          at-
pymeasure.instruments.novanta, 491
                                                     tribute), 292
pymeasure.instruments.oxfordinstruments,
                                             multidrop_capability
                                                                                     (pymea-
                                                     sure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38
pymeasure.instruments.parker, 505
                                                     property), 591
pymeasure.instruments.pendulum, 507
                                             multidrop_capability
                                                                                      (pymea-
```

	sure.instruments.tdk.tdk_gen80_65.TDK_Gen80_property), 595		or (pymeasure.instruments.keithley.K property), 366	eithley2700
N			or (pymeasure.instruments.keithley.K property), 369	eithley2750
NA (	pymeasure.instruments.hp.hp856Xx.StatusRegister attribute), 303		or (pymeasure.instruments.keithley.K property), 374	eithley6221
ND287	(class in pymeasure.instruments.heidenhain), 259		or (pymeasure.instruments.keithley.K	eithley6517B
Negat	ivePeak (pymea-	-	property), 380	
	sure.instruments.hp.hp856Xx.DetectionModes attribute), 302	1	or (pymeasure.instruments.keithley.K property), 402	
	urve() (pymeasure.display.widgets.dock_widget.Do method), 88	1	property), 436	
	urve() (pymeasure.display.widgets.image_widget.In method), 84	1	property), 418	
	urve() (pymeasure.display.widgets.plot_widget.Plot method), 85	1	property), 426	
new_c	urve() (pymeasure.display.widgets.tab_widget.TabV	w <del>q.q.x.t</del> _err	or (pymeasure.instruments.lecroy.LeC	CroyT3DSO1204
	method), 88	I	property), 464	
new_c	urve() (pymeasure.display.widgets.table_widget.Tat method), 90	I	property), 510	
next(	method), 79	1	or (pymeasure.instruments.signalreco property), 554	
next(	method), 80	Į.	or (pymeasure.instruments.signalreco property), 561	·
	error (pymeasure.instruments.aimtti.ld400p.LD400 property), 218	1	property), 568	
next_	error (pymeasure.instruments.andeenhagerling.AH. property), 229	I	property), 591	
	error (pymeasure.instruments.fwbell.FWBell5080 property), 258	1	or (pymeasure.instruments.tdk.tdk_ge property), 596	
next_	error (pymeasure.instruments.generic_types.SCPIM property), 108	I	property), 600	
	error (pymeasure.instruments.Instrument property), 104	1	or (pymeasure.instruments.texio.Texio property), 623	oPSW360L30
next_	error (pymeasure.instruments.keithley.Keithley2000 property), 327	gnext_set	point() sure.instruments.temptronic.ATS545	(pymea- method),
next_	error (pymeasure.instruments.keithley.Keithley2182		521	
	property), 396	next_set		(рутеа-
	error (pymeasure.instruments.keithley.Keithley2200 property), 333	(	sure.instruments.temptronic.ATSBase 517	
	error (pymeasure.instruments.keithley.Keithley2260 property), 336	1	method), 412	
	error (pymeasure.instruments.keithley.Keithley2281 property), 339	(	attribute), 302	
	error (pymeasure.instruments.keithley.Keithley2306 property), 342	C	attribute), 302	
next_	error (pymeasure.instruments.keithley.Keithley2400 property), 347	gNextRigh	t (pymeasure.instruments.hp.hp856X: attribute), 302	x.PeakSearchMode
next_	error (pymeasure.instruments.keithley.Keithley2450	gnicam_ad	ditional_bits	(рутеа-
	property), 355	S	sure.instruments.rohdeschwarz.sfm.SF	FM
next_	error (pymeasure.instruments.keithley.Keithley2510		property), 531	,
	property), 360		dio_frequency	(pymea-
next_	error (pymeasure.instruments.keithley.Keithley2600 property), 363	-	sure.instruments.rohdeschwarz.sfm.SF property), 531	' IVI

nicam_audio_volume (pymea-	tribute), 293
sure.instruments.rohdeschwarz.sfm.SFM	normalized_reference_level (pymea-
property), 532	sure.instruments.hp.hp856Xx.HP856Xx at-
nicam_bit_error_enabled (pymea-	tribute), 293
sure.instruments.rohdeschwarz.sfm.SFM	normalized_reference_position (pymea-
	sure.instruments.hp.hp856Xx.HP856Xx at-
property), 532	
nicam_bit_error_rate (pymea-	tribute), 294
sure.instruments.rohdeschwarz.sfm.SFM	not_at_temperature() (pymea-
property), 532	sure.instruments.temptronic.ATSBase method),
nicam_carrier_enabled (pymea-	617
sure.instruments.rohdeschwarz.sfm.SFM	nozzle_air_flow_rate (pymea-
property), 532	sure.instruments.temptronic.ATSBase prop-
nicam_carrier_frequency (pymea-	erty), 617
sure.instruments.rohdeschwarz.sfm.SFM	nplc (pymeasure.instruments.hp.HP34401A property),
property), 532	264
nicam_carrier_level (pymea-	nplc (pymeasure.instruments.keithley.KeithleyDMM6500
sure.instruments.rohdeschwarz.sfm.SFM	property), 389
property), 532	${\tt nplc} \ (pymeasure. instruments. keithley. keithley DMM 6500. Scanner Card 2000 for the properties of the propertie$
nicam_control_bits (pymea-	property), 394
sure.instruments.rohdeschwarz.sfm.SFM	null_operation_enabled (pymea-
property), 532	sure.instruments.advantest.advantestR624X.SMUChannel
$\verb+nicam_data+ (pymeasure. instruments. rohdeschwarz. sfm. SF+ (pymeasure. instruments. rohdeschwarz. sfm. sfm. sfm. sfm. sfm. sfm. sfm. sfm$	FM property), 137
property), 532	num_ch (pymeasure.instruments.activetechnologies.AWG401x_AWG
nicam_intercarrier_frequency (pymea-	property), 113
sure.instruments.rohdeschwarz.sfm.SFM	num_dch (pymeasure.instruments.activetechnologies.AWG401x_AWG
property), 532	property), 113
	num_points (pymeasure.instruments.agilent.Agilent4294A
sure.instruments.rohdeschwarz.sfm.SFM	property), 172
property), 531	number_of_channels (pymea-
nicam_mode (pymeasure.instruments.rohdeschwarz.sfm.SF	**
property), 532	property), 240
	number_of_channels (pymea-
sure.instruments.rohdeschwarz.sfm.SFM	sure.instruments.inficon.sqm160.SQM160
	property), 320
property), 532 nicam_source (pymea-	
= :	number_of_points (pymea- sure.instruments.anritsu.anritsuMS464xB.MeasurementChannel
sure.instruments.rohdeschwarz.sfm.SFM	
property), 533	property), 244
= :	number_of_ports (pymea-
sure.instruments.rohdeschwarz.sfm.SFM	sure.instruments.anritsu.AnritsuMS464xB
property), 533	property), 241
noise (pymeasure.instruments.rigol.rigol_dg800.VoltageC	
property), 525	sure.instruments.anritsu.anritsuMS464xB.MeasurementChannel
nominal_level (pymea-	property), 244
sure.instruments.rohdeschwarz.fsseries.FSW	number_readings (pymea-
property), 542	sure.instruments.hp.HP3437A property),
NONE (pymeasure.instruments.hp.hp856Xx.StatusRegister	267
attribute), 303	Nxds (class in pymeasure.instruments.edwards), 253
${\tt Normal}\ (pymeasure.instruments.hp.hp856Xx.DetectionModel (pymeasure.instruments.$	des
attribute), 302	0
normal_channel (pymea-	OCP_enabled (pymeasure.instruments.hp.HP6632A
sure.instruments.rohdeschwarz.sfm.SFM	property), 317
property), 533	Off (pymeasure.instruments.hp.hp856Xx.DemodulationMode
normalize_trace_data_enabled (pymea-	attribute), 301
sure.instruments.hp.hp856Xx.HP856Xx at-	wentome j, 501

off_tir	${\sf ne}$ (pymeasure.instruments.kuhneelectronic.Kusg $2$		(рутеа-
	property), 442	sure.instruments.agilent.agilent4284.	A.Agilent4284ACorrection
offset	(pymeasure.instruments.agilent.Agilent33220A	property), 200	
	property), 173	open_file_externally()	(рутеа-
offset	(pymeasure.instruments.agilent.Agilent33500 property), 176	sure.display.windows.managed_wind method), 93	low.ManagedWindowBase
offset	(pymeasure.instruments.agilent.agilent33500.Agil		(рутеа-
OIIDCC	property), 179	sure.instruments.keithley.Keithley270	
offset	(pymeasure.instruments.agilent.agilent4156.VARI		
OIISCC	property), 170	operating_hours	(рутеа-
offset		sure.instruments.thyracont.smartline	* *
UIISEL	erty), 263	property), 631	_v2.Smartimev2
offcot	(pymeasure.instruments.hp.HP437B property),		n HP/137R
	313	property), 313	_
offset	(pymeasure.instruments.hp.HP8116A property), 272	operating_mode (pymeasure.instruments.hp property), 273	o.HP8116A
offset	pymeasure.instruments.keysight.Keysight81160A	operating_mode	(рутеа-
	property), 436	sure.instruments.kepco.KepcoBOP36	512
offset	(pymeasure.instruments.tektronix.afg3152c.AFG3		
	property), 599	operating_mode	(рутеа-
offset	(pymeasure.instruments.teledyne.teledyne_oscillo.		property),
	property), 609	520	1 1 277
offset	(pymeasure.instruments.teledyne.teledyneT3AFG.	Si <b>one</b> k@kawowlenable reg	(рутеа-
	property), 602	sure.instruments.rohdeschwarz.sfm.S	• •
offset	_compensated (pymea-	property), 533	
	sure.instruments.keithley.KeithleyDAQ6510	operation_event_enabled	(рутеа-
	property), 402	sure.instruments.keithley.Keithley622	
offset_	current (pymeasure.instruments.srs.SR570	erty), 374	
	property), 574	operation_events	(рутеа-
offset_	_current_enabled (pymea-	sure.instruments.keithley.Keithley622	21 prop-
	sure.instruments.srs.SR570 property), 574	erty), 374	
offset_	= :	operation_hours	(рутеа-
	sure.instruments.srs.SR570 property), 574	sure.instruments.mksinst.mks974b.M	KS974B
offset_	_enabled (pymeasure.instruments.hp.HP437B	property), 475	
	property), 313	operation_mode	(pymea-
offset_	_voltage	sure.instruments.tcpowerconversion.	CXN
	sure.instruments.attocube.anc300.Axis prop-	property), 587	
	erty), 248	operation_register	(pymea-
open()	(pymeasure.instruments.keithley.Keithley2750	sure.instruments.advantest.advantest	R624X.SMUChannel
	method), 369	property), 138	
open_a	11() (pymeasure.instruments.keithley.Keithley275	$70\mathrm{options}$ (pymeasure.instruments.agilent. $A$ gile	entE5270B
	method), 369	property), 214	
open_a	ll_channels() (pymea-	options (pymeasure.instruments.aimtti.ld400	)p.LD400P
-	sure.instruments.keithley.Keithley2700	property), 218	•
	method), 366	options (pymeasure.instruments.andeenhager	rling.AH2700A
open_cl	nannel() (pymea-	property), 229	
• –	sure.instruments.keithley.KeithleyDAQ6510	options (pymeasure.instruments.fwbell.F	WBell5080
	method), 402	property), 258	
open_cl		options (pymeasure.instruments.generic_type	s.SCPIMixin
- F C	sure.instruments.keithley.Keithley2700 prop-	property), 108	
	erty), 366	options (pymeasure.instruments.hp.HP811)	6A pron-
open cl	nannels() (pymea-	erty), 273	or high
open_ci	sure.instruments.keithley.KeithleyDAQ6510	options (pymeasure.instruments.hp.HP875.	3E prop-
	method), 402	erty), 308	- Prop

options	(pymeasure.instruments.Instrument property), 104	_	(pymeasure.instruments.teledyne.TeledyneT3AFG property), 600
_	(pymeasure.instruments.keithley.Keithley2000 property), 327	options	(pymeasure.instruments.texio.TexioPSW360L30 property), 623
	(pymeasure.instruments.keithley.Keithley2182		requency (pymea-
	property), 396	01011_1	sure.instruments.hp.hp856Xx.HP856Xx at-
	(pymeasure.instruments.keithley.Keithley2200		tribute), 292
	property), 333	outnut	(pymeasure.instruments.agilent.Agilent33220A
	(pymeasure.instruments.keithley.Keithley2260B	output	
_	* * *		property), 173
	property), 336	output	(pymeasure.instruments.agilent.Agilent33500
	(pymeasure.instruments.keithley.Keithley2281S		property), 176
	property), 339		pymeasure.instruments.agilent.agilent33500.Agilent33500Channe
	(pymeasure.instruments.keithley.Keithley2306		property), 179
	property), 342		pymeasure.instruments.anritsu.AnritsuMG3692C
	(pymeasure.instruments.keithley.Keithley2400		property), 230
	property), 347	_	pymeasure.instruments.keysight.Keysight81160A
	(pymeasure.instruments.keithley.Keithley2450		property), 437
	property), 355	_	$pymeasure.instruments.lakeshore.lakeshore\_base.LakeShoreHeate$
_	(pymeasure.instruments.keithley.Keithley2510		property), 455
	property), 360	output	(pymeasure.instruments.srs.SR510 property),
options	(pymeasure.instruments.keithley.Keithley2600		573
	property), 363	output_	1 (pymeasure.instruments.lakeshore.LakeShore331
options	(pymeasure.instruments.keithley.Keithley2700		attribute), 448
	property), 367	output_	1 (pymeasure.instruments.lakeshore.LakeShore3xx
	(pymeasure.instruments.keithley.Keithley2750	-	attribute), 450
	property), 369	output	1 (pymeasure.instruments.razorbill.razorbillRP100
	(pymeasure.instruments.keithley.Keithley4200		property), 521
	property), 405		2 (pymeasure.instruments.lakeshore.LakeShore331
	(pymeasure.instruments.keithley.Keithley6221		attribute), 448
	property), 374	output	2 (pymeasure.instruments.lakeshore.LakeShore3xx
	(pymeasure.instruments.keithley.Keithley6517B	output_	attribute), 450
	property), 380	outnut	2 (pymeasure.instruments.razorbill.razorbillRP100
	(pymeasure.instruments.keithley.KeithleyDAQ651		property), 521
	property), 402		3 (pymeasure.instruments.lakeshore.LakeShore3xx
	* * * * · ·		
	(pymeasure.instruments.keysight.Keysight81160A		attribute), 450
	property), 436		4 (pymeasure.instruments.lakeshore.LakeShore3xx
	(pymeasure.instruments.keysight.KeysightE36312.		attribute), 450
	property), 418		all_measurements() (pymea-
	(pymeasure.instruments.keysight.KeysightE3631A		sure.instruments.advantest.advantestR624X.SMUChannel
	property), 426		method), 137
_	(py measure. instruments. lecroy. LeCroy T3DSO 120)	_	* *
	property), 464		sure.instruments.srs.SR830 method), 577
_	(pymeasure.instruments.proterial.ROD4 prop-	output_	
	erty), 510		sure.instruments.advantest.advantestR624X.SMUChannel
options	(pymeasure.instruments.signalrecovery.DSP7225		property), 139
	property), 554	output_	enabled ( <i>pymea-</i>
options	(pymeasure.instruments.signalrecovery.DSP7265		sure.instruments.agilent.AgilentE5062A
	property), 561		property), 157
	(pymeasure.instruments.srs.ldc500series.LDC500		A A
_	property), 568	_	sure.instruments.aimtti.aimttiPL.PLChannel
	(pymeasure.instruments.tdk.tdk_gen40_38.TDK_0		
_	property), 591	output_	A A
	(pymeasure.instruments.tdk.tdk_gen80_65.TDK_0	_	
_	property), 596		property), 255
	* * *//		<b>.</b>

<pre>output_enabled (pymeasure.instruments.hp.HP6632</pre>		property), 176	
property), 318		load (pymeasure.instruments.agilent.ag	gilent33500.Agilent33500C
output_enabled (pymeasure.instruments.hp.HP8116		property), 179	** 011604
property), 273		load (pymeasure.instruments.keysight.	Keysight81160A
output_enabled (pymeasure.instruments.hp.HP8657		property), 437	,
property), 306		low_grounded	(pymea-
output_enabled (pymeasure.instruments.hp.HP8753	E	sure.instruments.keithley.Keithley6221	prop-
property), 308		erty), 374	
	a- output_		(pymea-
sure.instruments.keithley.keithley2200.PSCha	nnei	sure.instruments.keithley.Keithley2400	) prop-
property), 334	~ <b>0+m</b> +	erty), 347	(77,144.0.7
- · · · · · · · · · · · · · · · · · · ·	a- output_	_	(pymea-
sure.instruments.keithley.Keithley2260B		sure.instruments.keysight.KeysightN77	7700
property), 336	a 011+n11+	property), 412	(m)maa
- · · · · · · · · · · · · · · · · · · ·	a- output_	_	(pymea-
sure.instruments.kepco.KepcoBOP3612		sure.instruments.keysight.KeysightN77	7700
property), 407	~ <b>0+m</b> +	property), 412	(77,144.0.7
		trigger_on_external()	(pymea-
sure.instruments.keysight.keysightE36312A.V	oitageCnanne		,
property), 420		method), 347	(
	a- output_	trigger_on_external()	(pymea-
sure.instruments.keysight.KeysightE3631A		sure.instruments.keithley.Keithley6221	!
property), 427		method), 374	,
	a- output_		(рутеа-
sure.instruments.keysight.keysightE3631A.Vol	ltageChannel		s prop-
property), 429		erty), 248	,
	a- output_	_	(рутеа-
sure.instruments.keysight.KeysightN7776C		sure.instruments.fakes.SwissArmyFake	e prop-
property), 412		erty), 107	,
	a- output_	_	(pymea-
sure.instruments.rigol.rigol_dg800.VoltageCh	iannel	sure.instruments.rohdeschwarz.sfm.SF	M
property), 525		property), 533	,
	a- outputs		(pymea-
sure.instruments.rohdeschwarz.hmp.HMP404	0	sure.instruments.ni.virtualbench.Virtua	alBench.PowerSupply
property), 543		property), 489	
output_enabled (pymed	a- OutputT		pymea-
sure.instruments.spellmanhv.SpellmanXRV		sure.instruments.advantest.advantestR	624X),
property), 565		140	,
	a- over_vo		(pymea-
sure.instruments.tdk.tdk_gen40_38.TDK_Gen	40_38	sure.instruments.tdk.tdk_gen40_38.TD	0K_Gen40_38
property), 591		property), 591	,
	a- over_vo	_	(pymea-
sure.instruments.tdk.tdk_gen80_65.TDK_Gen	80_65	sure.instruments.tdk.tdk_gen80_65.TD	0K_Gen80_65
property), 596		property), 596	
		oltage_limit	(pymea-
sure.instruments.teledyne.teledyneT3AFG.Sig	nalChannel		property),
property), 602		318	
output_enabled (pymed	a- OxfordI	nstrumentsBase (class in	рутеа-
sure.instruments.texio.TexioPSW360L30		sure.instruments.oxfordinstruments.ba	se),
property), 623		493	
	a- OxfordV	· ·	рутеа-
sure.instruments.activetechnologies.AWG401x	c.ChannelAF0	· ·	se),
property), 115		494	
<pre>output_load(pymeasure.instruments.agilent.Agilent3.</pre>	3500		

P	parse() (pymeasure.experiment.results.Results method),
<pre>pandas_column_count()</pre>	_ 72
sure.display.widgets.table_widget.PandasModeli	$plantimeter_{alberts}$ (pymea-
method), 89	sure.aispiay.wiageis.pioi_frame.i toti rame
<pre>pandas_column_count()</pre>	_ method), 84
sure.display.widgets.table_widget.PandasModeli	elBypanse_columns() (pymea-
method), 89	sure.experimeni.procedure.Frocedure static
pandas_column_count() (pymea-	_ method), 66
sure.display.widgets.table_widget.PandasModell	elByRayse_header() (pymeasure.experiment.results.Results static method), 73
method), 90	names ctroam() (numargum consumtor Consumtor
pandas_row_count() (pymea-	J D (1
sure.display.widgets.table_widget.PandasModell	part_number(pymeasure.instruments.mksinst.mks974b.MKS974B
method), 89	mun autu) 475
pandas_row_count() (pymea-	elBy <mark>column</mark> t 1ter (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38
sure.display.widgets.table_widget.PandasModell method), 89	property), 591
pandas_row_count() (pymea-	5-11 (
sure.display.widgets.table_widget.PandasModeli	. \ 506
method), 90	pattern_down (pymea-
PandasModelBase (class in pymea-	sura instruments attacube and 300 Avis nran-
sure.display.widgets.table_widget), 88	erty), 248
PandasModelByColumn (class in pymea-	pattern_up(pymeasure.instruments.attocube.anc300.Axis
sure.display.widgets.table_widget), 89	property), 248
PandasModelByRow (class in pymea-	pause() (pymeasure.instruments.agilent.agilentB1500.AgilentB1500
sure.display.widgets.table_widget), 90	method), 185
parallel_meas (pymea-	pause() (pymeasure.instruments.ami.AMI430 method),
sure.instruments.agilent.agilentB1500.AgilentB1	
property), 186	pause_scan() (pymeasure.instruments.srs.SR830
Parameter (class in pymeasure.experiment.parameters),	method), 577
69	pb_desc (pymeasure.instruments.hp.HP3437A at-
<pre>parameter (pymeasure.display.inputs.Input property),</pre>	tribute), 267
77	pd (pymeasure.instruments.srs.ldc500series.LDC500Series
$\verb"parameter" (pymeasure. instruments. a gilent. a gilent E 5062 A structure of the parameter of the parame$	2A.VNATrace <sup>attribute</sup> ), 568
property), 161	peak_excursion (pymea-
<pre>parameter_DAT1 (pymeasure.instruments.srs.SR860</pre>	g sure.instruments.hp.hp856Xx.HP856Xx at-
property), 582	tribute), 291
<pre>parameter_DAT2 (pymeasure.instruments.srs.SR860</pre>	peak_preselector() (pymea-
property), 582	sure.instruments.hp.HP8561B method), 299
parameter_DAT3 (pymeasure.instruments.srs.SR860	peak_search(pymeasure.instruments.anritsu.AnritsuMS9710C
property), 582	property), 232
parameter_DAT4 (pymeasure.instruments.srs.SR860	9 PeakSearchMode (class in pymea-
property), 582	sure.instruments.hp.hp856Xx), 302
<pre>parameter_objects()</pre>	227
sure.experiment.procedure.Procedure method),	property), 327
66	period (pymeasure.instruments.keithley.KeithleyDMM6500
parameter_values() (pymea-	
sure.experiment.procedure.Procedure method),	period_aperature (pymea-
66	sure.instruments.keithley.Keithley2000 prop- erty), 327
parameters_are_set() (pymea-	
sure.experiment.procedure.Procedure method),	sure.instruments.keithley.KeithleyDMM6500
66	
parent() (pymeasure.display.widgets.sequencer_widget.S	t.SequencerTreeMottel <sup>y, 367</sup> period_digits (pymea-
method), 86	and in the second of the ideal of Weight and 2000
ParkerGV6 (class in pymeasure.instruments.parker), 505	) sure.mism umemis.nemme y.nemme y2000 prop

erty), 327	sure.instruments.keysight.Keysight81160A
period_digits (pymea-	method), 437
sure. instruments. keithley. Keithley DMM 6500	PhysicalParameter (class in pymea-
property), 389	sure.experiment.parameters), 70
period_reference (pymea-	piezo (pymeasure.instruments.thyracont.smartline_v2.VSM
sure.instruments.keithley.Keithley2000 prop-	attribute), 632
erty), 327	piezo (pymeasure.instruments.thyracont.smartline_v2.VSP
period_relative (pymea-	attribute), 632
sure.instruments.keithley.KeithleyDMM6500	piezo (pymeasure.instruments.thyracont.smartline_v2.VSR
property), 389	attribute), 632
period_relative_enabled (pymea-	piezo_pressure (pymea-
sure.instruments.keithley.KeithleyDMM6500	sure.instruments.mksinst.mks974b.MKS974B
property), 389	property), 475
period_threshold (pymea-	ping() (pymeasure.instruments.hcp.TC038D method), 261
sure.instruments.keithley.Keithley2000 property), 327	
period_threshold (pymea-	ping() (pymeasure.instruments.tcpowerconversion.CXN method), 587
sure.instruments.keithley.KeithleyDMM6500	pirani (pymeasure.instruments.thyracont.smartline_v2.VSH
property), 389	attribute), 632
period_threshold_auto_enabled (pymea-	pirani (pymeasure.instruments.thyracont.smartline_v2.VSR
sure.instruments.keithley.KeithleyDMM6500	attribute), 632
property), 389	pirani_pressure (pymea-
persistent_field (pymea-	sure.instruments.mksinst.mks974b.MKS974B
sure.instruments.oxfordinstruments.IPS120_10	property), 475
property), 500	PL068P (class in pymeasure.instruments.aimtti.aimttiPL),
phase (pymeasure.instruments.activetechnologies.AWG40	1x.ChannelAFG
property), 115	PL155P (class in pymeasure.instruments.aimtti.aimttiPL),
phase (pymeasure.instruments.agilent.Agilent33500	216
property), 176	PL303P (class in pymeasure.instruments.aimtti.aimttiPL),
$phase ({\it pymeasure.instruments.agilent.agilent33500.Ag$	
property), 179	PL303QMDP (class in pymea-
phase (pymeasure.instruments.ametek.Ametek7270 prop-	sure.instruments.aimtti.aimttiPL), 217
erty), 221	PL303QMTP (class in pymea-
phase (pymeasure.instruments.keysight.Keysight81160A	sure.instruments.aimtti.aimttiPL), 217
property), 437	PL601P (class in pymeasure.instruments.aimtti.aimttiPL), 216
phase (pymeasure.instruments.signalrecovery.DSP7225 property), 554	placeholder_names() (pymea-
phase (pymeasure.instruments.signalrecovery.DSP7265	sure.experiment.procedure.Procedure class
property), 561	method), 66
phase (pymeasure.instruments.srs.SR510 property), 573	placeholder_objects() (pymea-
phase (pymeasure.instruments.srs.SR830 property), 577	sure.experiment.procedure.Procedure method),
phase (pymeasure.instruments.srs.SR860 property), 582	66
phase() (pymeasure.instruments.agilent.Agilent8722ES	PlaceholderCompleter (class in pymea-
method), 152	sure.display.widgets.filename_widget), 83
	/P4BasehaluseliAff@measure.instruments.aimtti.aimttiPL),
property), 115	215
$\verb"phase_min" (pymeasure. instruments. active technologies. AW) \\$	TO4OUpGhansuetAFiGtruments.agilent.agilentB1500.ADCMode
property), 115	attribute), 195
$\verb phase_shift  (pymeasure.instruments.kuhneelectronic.Kuhnee$	s <b>FLLh</b> amel (class in pymea-
property), 442	sure.instruments.aimtti.aimttiPL), 216
phase_sync() (pymea-	plot() (pymeasure.experiment.experiment.Experiment
sure.instruments.agilent.Agilent33500 method),	method), 64
177	plot_live() (pymeasure.experiment.experiment.Experiment
phase_sync() (pymea-	method), 64

PlotFr	ame	(class	in	рутеа-		492	
	sure.displ	ay.widgets.plot	_frame), 84		power(p	ymeasure.instruments.oxfordinstrum	ents.mercuryitc.Heater
Plotte	r (class in p	ymeasure.disp	lay.plotter), 8	31		property), 503	
Plotte	rWindow	(class	in	рутеа-	power(p	y me a sure. in struments. siglent technology and the sum of the	ogies.siglent_spdbase.SPDCh
	sure.displ	ay.windows.plo	otter_window	), 94		property), 547	
PlotWi	•	(class ay.widgets.plot	in widget), 84	рутеа-	power(p	rymeasure.instruments.srs.ldc500sert property), 570	ies.LDC500SeriesPD
PM6669	_	measure.instru	-	3), 506	power	(pymeasure.instruments.tcpowercon	version.CXN
		re.instruments.o				property), 587	
_	property).		v		power	(pymeasure.instruments.texio.Texio.	PSW360L30
points	(pymeasure	instruments.ag	gilent.agilent	4156.VAR2		property), 623	
	property).	, 170			power(p	ymeasure.instruments.thorlabs.Thor	labsPM100USB
points	_in_buffe	r		(pymea-		property), 626	
	sure.instr	uments.keithley	.KeithleyDM	M6500	power(p	ymeasure.instruments.toptica.ibeam	smart.DriverChannel
	property).					property), 635	
polari			.danfysik.Da	nfysik8500	power(p	ymeasure.instruments.toptica.ibeam	smart.IBeamSmart
	property).					property), 634	
pop_er		sure.instrumen	ts.agilent.Agi	lentE5062A	power_c		(рутеа-
_	method),					sure.instruments.anritsu.AnritsuMS	2090A
Port	*	lass	in	pymea-		property), 236	,
		uments.anritsu.	anritsuMS40	4xB),	power_e		(pymea-
	245		anta la aidamla a	: ND297		sure.instruments.mksinst.mks937b.	PressureCnannei
positi	property).	easure.instrume	ть.пешети	III.ND207	power_f	property), 474	(рутеа-
nositi		, 200 sure.instrument	ts nownort os	n300 Aris	bower_r	sure.instruments.kuhneelectronic.K	
positi	property).		is.newport.es	ρ500.71λιδ		property), 442	usg2+3_25011
positi		easure.instrum	ents narker P	arkerGV6	nower 1	evel (pymeasure.instruments.anrits	u anritsuMS464xR Port
posici	property).		erris.parrier.1	arner o r o	power_r	property), 245	
positi	on_error	, 0 00		(pymea-	power 1	imit (pymeasure.instruments.srs.lde	c500series.LDC500SeriesPD
•		uments.parker.l	ParkerGV6	4.5	. –	property), 570	
	505	1	•	1 277	power_l	.imit (pymeasure.instruments.tcpow	erconversion.CXN
Positi	vePeak			(pymea-		property), 587	
	sure.instr	uments.hp.hp85	56Xx.Detectio	onModes	power_l	imits	(pymea-
	attribute).	, 302				sure.instruments.spellmanhv.Spellm	nanXRV
power		ure.instrument	s.agilent.Agil	ent8257D		property), 565	
	property).					on_clear	(рутеа-
power(	pymeasure.i property)	_	lent.agilentE	5062A.VNA	Channel	sure.instruments.advantest.advantest.property), 125	stR624X.AdvantestR624X
power (		instruments.aja	.DCXS prope	erty), 219	power_r	range (pymeasure.instruments.ipgph	otonics.yar.YAR
_		instruments.and		•	-	property), 322	·
	erty), 227				power_r	reference_enabled	(pymea-
power (	pymeasure.	instruments.an	ritsu.Anritsul	MG3692C		sure.instruments.hp.HP437B prope	rty), 313
	property).				power_r		(pymea-
power	(pymeasure 313	e.instruments.h	p.HP437B	property),		sure.instruments.kuhneelectronic.K property), 443	usg245_250A
power	(pymeasure	.instruments.hp	o.HP8753E	property),	power_s	setpoint	(pymea-
	308					sure.instruments.ipgphotonics.yar.Y	YAR prop-
power	(pymeas	ure.instruments	s.ipgphotonic	s.yar.YAR		erty), 322	
	property).				_	setpoint	(рутеа-
power(		instruments.kei	thley.keithley	2200.PSCh	annel	sure.instruments.kuhneelectronic.K	usg245_250A
	property).		1 1.11 **	1 00 505		property), 443	
power		re.instruments.i	keithley.Keith	uey2260B	power_s	setpoint	(pymea-
	property).		F (0			sure.instruments.novanta.Fpu60	property),
power (	pymeasure.i	instruments.nov	vanta.F pu00 j	property),		492	

power_setpoint	(рутеа-		property), 628	
sure.instruments.srs.ldc500series.	LDC500Series	PpDressur	e (pymeasure.instruments.thyracont.s	smartline_v2.SmartlineV2
property), 570			property), 631	
preamp (pymeasure.instruments.anritsu.Anr	itsuMS2090A	Pressur	eChannel (class in	рутеа-
property), 236			sure.instruments.mksinst.mks937b),	1.5
preemphasis_enabled	(рутеа-	preview	_widget()	(pymea-
sure.instruments.rohdeschwarz.sfr	4.0	-	sure.display.widgets.plot_widget.Plc	A 2
property), 537	souna_enam	101	method), 85	, mager
preemphasis_time	(nymea-	nreview	_widget()	(рутеа-
sure.instruments.rohdeschwarz.sfr	* *	-	_wragee() sure.display.widgets.tab_widget.Tab	4.5
property), 537	п.50ина_спат	ici	method), 88	mugei
preheat (pymeasure.instruments.spellmank	w spallmanVDI	/ Filowiout		(mymag
property), 567	іч. эрентапак ч	.ршемивем		(pymea-
	D. G C		sure.display.widgets.table_widget.To	able włagei
prepare() (pymeasure.display.curves	s.BujjerCurve		method), 91	,
method), 76	,	previou	s_step()	(pymea-
preselector_dac_number	(pymea-		sure.instruments.keysight.KeysightN	1///6C
sure.instruments.hp.HP8561B	property),	_	method), 412	
299		probe_a	ttenuation	(рутеа-
preset() (pymeasure.instruments.hp.HP4.313	37B method),		sure.instruments.teledyne.teledyne_oproperty), 609	oscilloscope.TeledyneOscillos
<pre>preset() (pymeasure.instruments.hp.hp856</pre>	6Xx.HP856Xx	probe_t	ype (pymeasure.instruments.lakesho	re.LakeShore421
method), 274		_	property), 452	
**	al.Racal1992	Procedu	re (class in pymeasure.experiment.)	procedure),
method), 520			65	, ,
preset_1 (pymeasure.instruments.tcpowere	conversion CXN	/product	name	(рутеа-
attribute), 585		Produce	sure.instruments.thyracont.smartline	
preset_2 (pymeasure.instruments.tcpowere	conversion CXN	J	property), 631	e_v2.5martimev2
attribute), 585	onversion.CAI		_sweep()	(pymea-
preset_3 (pymeasure.instruments.tcpowere	conversion CVA		_sure.instruments.oxfordinstruments	4.5
attribute), 586	onversion.CAN	V	method), 496	11 C 30 3
	omnamian CVA	/Dwologi		taus) 52
preset_4 (pymeasure.instruments.tcpowerd	onversion.CXN			
attribute), 586	· ava		ional_band	(pymea-
<pre>preset_5 (pymeasure.instruments.tcpowerd</pre>	conversion.CXN	V	sure.instruments.oxfordinstruments.	11C503
attribute), 586		_	property), 496	
<pre>preset_6 (pymeasure.instruments.tcpowerd</pre>	conversion.CXN	/protect		(рутеа-
attribute), 586			sure.instruments.hp.hp856Xx.HP856	6Xx at-
<pre>preset_7 (pymeasure.instruments.tcpowere</pre>	conversion.CXN		tribute), 279	
attribute), 586		Protoco	1Adapter (class in pymeasure.adapt	ters), 58
<pre>preset_8 (pymeasure.instruments.tcpowere</pre>	conversion.CXN	√ps	(pymeasure.instruments.keithley.Kei	ithley2281S
attribute), 586			attribute), 337	
<pre>preset_9 (pymeasure.instruments.tcpowere</pre>	conversion.CXN	VPS120_1	0 (class in	рутеа-
attribute), 586			sure.instruments.oxfordinstruments)	, 504
<pre>preset_slot (pymeasure.instruments.tcpo)</pre>	werconversion.	Œ <b>%</b> ℃hann		рутеа-
PresetChannel (class in	рутеа-	nseudo	scanner_enabled	(pymea-
sure.instruments.tcpowerconversion	1 2	pscuuo_	sure.instruments.keithley.KeithleyDl	
588	m.iccan),		property), 390	WIW0500
	ulan 27h Dunan			(22242.2.2.2
pressure (pymeasure.instruments.mksinst.i	nks95/v.Pressi	n pasun <u>u</u> n man		(pymea-
property), 474	1 0741 147200	740	sure.instruments.novanta.Fpu60	property),
pressure (pymeasure.instruments.mksinst.i	nks9/4b.MKS9		492	
property), 475	01116	ptwDIAM	ENTOR (class in pymeasure.instru	ments.ptw),
pressure (pymeasure.instruments.ptw.ptwl	DIAMENTOR		512	
property), 513		_	OS (class in pymeasure.instruments.p	
pressure (pymeasure.instruments.thyracon	t.smartline v1.	Somotistino	Vmeasure.instruments.rigol.rigol_de8	800.VoltageChannel

```
property), 525
                                                              sure.instruments.agilent.agilent33500.Agilent33500Channel
pulse_dutycycle
                                                             property), 180
                                           (рутеа-
        sure.instruments.agilent.Agilent33220A
                                                    pulse_transition
                                                                                                (pymea-
        property), 173
                                                             sure.instruments.keysight.Keysight81160A
pulse_dutycycle
                                           (рутеа-
                                                             property), 437
        sure.instruments.agilent.Agilent33500
                                             prop-
                                                    pulse_width(pymeasure.instruments.agilent.Agilent33220A
        erty), 177
                                                             property), 173
pulse_dutycycle
                                           (pymea- pulse_width(pymeasure.instruments.agilent.Agilent33500
        sure.instruments.agilent.agilent33500.Agilent33500Channelproperty), 177
                                                    pulse_width(pymeasure.instruments.agilent.agilent33500.Agilent33500C
        property), 179
pulse_dutycycle
                                           (рутеа-
                                                             property), 180
                                                    pulse_width
        sure.instruments.keysight.Keysight81160A
                                                                      (pymeasure.instruments.hp.HP8116A
                                                             property), 273
        property), 437
pulse_frequency
                                           (рутеа-
                                                    pulse_width(pymeasure.instruments.keysight.Keysight81160A
        sure.instruments.agilent.Agilent8257D
                                                             property), 437
                                             prop-
                                                    pulse_width (pymeasure.instruments.kuhneelectronic.Kusg245_250A
pulse_hold(pymeasure.instruments.agilent.Agilent33220A
                                                             property), 443
        property), 173
                                                    pulse_width(pymeasure.instruments.rigol.rigol_dg800.VoltageChannel
pulse_hold(pymeasure.instruments.agilent.Agilent33500
                                                             property), 525
                                                    pymeasure.display.browser
        property), 177
pulse_hold(pymeasure.instruments.agilent.agilent33500.AgilemodiaDeChannel
        property), 179
                                                    pymeasure.display.console
pulse_hold(pymeasure.instruments.keysight.Keysight81160A module, 75
        property), 437
                                                    pymeasure.display.curves
pulse_input (pymeasure.instruments.agilent.Agilent8257D
                                                         module, 76
        property), 151
                                                     pymeasure.display.inputs
pulse_mode_enabled
                                           (рутеа-
                                                         module, 77
        sure.instruments.kuhneelectronic.Kusg245_250A pymeasure.display.listeners
                                                         module, 79
        property), 443
pulse_params
                                           (рутеа-
                                                    pymeasure.display.log
        sure.instruments.tcpowerconversion.CXN
                                                         module, 79
        property), 587
                                                    pymeasure.display.manager
                                                         module, 79
pulse_period
                                           (рутеа-
        sure.instruments.agilent.Agilent33220A
                                                    pymeasure.display.plotter
        property), 173
                                                         module, 81
pulse_period
                                                    pymeasure.display.thread
                                           (pymea-
        sure.instruments.agilent.Agilent33500
                                             prop-
                                                         module, 81
        erty), 177
                                                    pymeasure.display.widgets.browser_widget
pulse_period
                                           (рутеа-
                                                         module, 82
        sure.instruments.agilent.agilent33500.Agilent335000.Meansuale.display.widgets.directory_widget
        property), 179
                                                         module, 82
pulse_period
                                           (pymea-
                                                    pymeasure.display.widgets.dock_widget
        sure.instruments.keysight.Keysight81160A
                                                         module, 88
                                                     pymeasure.display.widgets.estimator_widget
        property), 437
                                                         module, 82
pulse_source
                                           (рутеа-
        sure.instruments.agilent.Agilent8257D
                                                    pymeasure.display.widgets.fileinput_widget
                                             prop-
        erty), 151
                                                         module, 82
pulse_transition
                                                    pymeasure.display.widgets.filename_widget
                                           (pymea-
        sure.instruments.agilent.Agilent33220A
                                                         module, 83
                                                    pymeasure.display.widgets.image_frame
        property), 173
pulse_transition
                                           (рутеа-
                                                         module, 83
        sure.instruments.agilent.Agilent33500
                                                    pymeasure.display.widgets.image_widget
                                             prop-
        erty), 177
                                                         module, 83
                                           (pymea- pymeasure.display.widgets.inputs_widget
pulse_transition
```

• • • • •	
module, 84	module, 215
pymeasure.display.widgets.log_widget	pymeasure.instruments.aja
module, 84	module, 218
<pre>pymeasure.display.widgets.plot_frame   module, 84</pre>	pymeasure.instruments.ametek module, 220
pymeasure.display.widgets.plot_widget	pymeasure.instruments.ami
module, 84	module, 222
pymeasure.display.widgets.results_dialog	pymeasure.instruments.anaheimautomation
module, 85	module, 224
pymeasure.display.widgets.sequencer_widget	pymeasure.instruments.anapico
module, 85	module, 226
<pre>pymeasure.display.widgets.tab_widget</pre>	pymeasure.instruments.andeenhagerling
module, 87	module, 227
<pre>pymeasure.display.widgets.table_widget</pre>	pymeasure.instruments.anritsu
module, 88	module, 230
<pre>pymeasure.display.windows.managed_dock_window</pre>	w pymeasure.instruments.attocube
module, 94	module, 245
<pre>pymeasure.display.windows.managed_image_wind</pre>	
module, 91	module, 248
<pre>pymeasure.display.windows.managed_window</pre>	pymeasure.instruments.comedi
module, 91	module, 110
pymeasure.display.windows.plotter_window	pymeasure.instruments.danfysik
module, 94	module, 249
pymeasure.experiment.experiment	pymeasure.instruments.deltaelektronika
module, 63	module, 252
<pre>pymeasure.experiment.listeners module, 64</pre>	pymeasure.instruments.edwards module, 253
module, 04	module, 233
pymeasure.experiment.parameters	pymeasure.instruments.eurotest
<pre>pymeasure.experiment.parameters   module, 67</pre>	pymeasure.instruments.eurotest module, 253
<pre>pymeasure.experiment.parameters    module, 67 pymeasure.experiment.procedure</pre>	<pre>pymeasure.instruments.eurotest    module, 253 pymeasure.instruments.fluke</pre>
<pre>pymeasure.experiment.parameters    module, 67 pymeasure.experiment.procedure    module, 65</pre>	<pre>pymeasure.instruments.eurotest    module, 253 pymeasure.instruments.fluke    module, 256</pre>
<pre>pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results</pre>	<pre>pymeasure.instruments.eurotest    module, 253 pymeasure.instruments.fluke    module, 256 pymeasure.instruments.fwbell</pre>
<pre>pymeasure.experiment.parameters     module, 67 pymeasure.experiment.procedure     module, 65 pymeasure.experiment.results     module, 72</pre>	<pre>pymeasure.instruments.eurotest     module, 253 pymeasure.instruments.fluke     module, 256 pymeasure.instruments.fwbell     module, 256</pre>
<pre>pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results</pre>	<pre>pymeasure.instruments.eurotest    module, 253 pymeasure.instruments.fluke    module, 256 pymeasure.instruments.fwbell</pre>
<pre>pymeasure.experiment.parameters     module, 67 pymeasure.experiment.procedure     module, 65 pymeasure.experiment.results     module, 72 pymeasure.experiment.workers</pre>	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp
<pre>pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71</pre>	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260
<pre>pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments</pre>	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260  pymeasure.instruments.heidenhain
<pre>pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95</pre>	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260  pymeasure.instruments.heidenhain    module, 259
pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results    module, 72  pymeasure.experiment.workers    module, 71  pymeasure.instruments    module, 95  pymeasure.instruments.activetechnologies    module, 111  pymeasure.instruments.aculight	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260  pymeasure.instruments.heidenhain    module, 259  pymeasure.instruments.hp    module, 261  pymeasure.instruments.inficon
pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results    module, 72  pymeasure.experiment.workers    module, 71  pymeasure.instruments    module, 95  pymeasure.instruments.activetechnologies    module, 111  pymeasure.instruments.aculight    module, 116	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260  pymeasure.instruments.heidenhain    module, 259  pymeasure.instruments.hp    module, 261  pymeasure.instruments.inficon    module, 319
pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results    module, 72  pymeasure.experiment.workers    module, 71  pymeasure.instruments    module, 95  pymeasure.instruments.activetechnologies    module, 111  pymeasure.instruments.aculight    module, 116  pymeasure.instruments.advantest	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260  pymeasure.instruments.heidenhain    module, 259  pymeasure.instruments.hp    module, 261  pymeasure.instruments.inficon    module, 319  pymeasure.instruments.ipgphotonics
pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results    module, 72  pymeasure.experiment.workers    module, 71  pymeasure.instruments    module, 95  pymeasure.instruments.activetechnologies    module, 111  pymeasure.instruments.aculight    module, 116  pymeasure.instruments.advantest    module, 117	pymeasure.instruments.eurotest    module, 253  pymeasure.instruments.fluke    module, 256  pymeasure.instruments.fwbell    module, 256  pymeasure.instruments.hcp    module, 260  pymeasure.instruments.heidenhain    module, 259  pymeasure.instruments.hp    module, 261  pymeasure.instruments.inficon    module, 319  pymeasure.instruments.ipgphotonics    module, 321
pymeasure.experiment.parameters    module, 67  pymeasure.experiment.procedure    module, 65  pymeasure.experiment.results    module, 72  pymeasure.experiment.workers    module, 71  pymeasure.instruments    module, 95  pymeasure.instruments.activetechnologies    module, 111  pymeasure.instruments.aculight    module, 116  pymeasure.instruments.advantest    module, 117  pymeasure.instruments.advantest.advantestR376	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipphotonics     module, 321  6709measure.instruments.keithley
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR376     module, 117	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ippphotonics     module, 321  670@measure.instruments.keithley     module, 322
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62	pymeasure.instruments.eurotest module, 253  pymeasure.instruments.fluke module, 256  pymeasure.instruments.fwbell module, 256  pymeasure.instruments.hcp module, 260  pymeasure.instruments.heidenhain module, 259  pymeasure.instruments.hp module, 261  pymeasure.instruments.inficon module, 319  pymeasure.instruments.ipgphotonics module, 321  6769measure.instruments.keithley module, 322  4%pymeasure.instruments.kepco
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR624     module, 139	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipgphotonics     module, 321  67000 measure.instruments.keithley     module, 322  4Xpymeasure.instruments.kepco     module, 407
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62     module, 139  pymeasure.instruments.agilent	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipgphotonics     module, 321  67©©measure.instruments.keithley     module, 322  4%pymeasure.instruments.kepco     module, 407  pymeasure.instruments.keysight
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62     module, 139  pymeasure.instruments.agilent     module, 148	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipgphotonics     module, 321  67000measure.instruments.keithley     module, 322  4%pymeasure.instruments.kepco     module, 407  pymeasure.instruments.keysight     module, 408
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62     module, 139  pymeasure.instruments.agilent     module, 148  pymeasure.instruments.agilent.agilent4156	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipphotonics     module, 321  6769measure.instruments.keithley     module, 322  4%pymeasure.instruments.kepco     module, 407  pymeasure.instruments.keysight     module, 408  pymeasure.instruments.kuhneelectronic
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62     module, 139  pymeasure.instruments.agilent     module, 148  pymeasure.instruments.agilent.agilent4156     module, 165	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipgphotonics     module, 321  6769measure.instruments.keithley     module, 322  4Xpymeasure.instruments.kepco     module, 407  pymeasure.instruments.keysight     module, 408  pymeasure.instruments.kuhneelectronic     module, 441
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62     module, 139  pymeasure.instruments.agilent     module, 148  pymeasure.instruments.agilent.agilent4156     module, 165  pymeasure.instruments.agilent.agilentB1500	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 260  pymeasure.instruments.hcp     module, 269  pymeasure.instruments.heidenhain     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ippphotonics     module, 321  676%measure.instruments.keithley     module, 322  4%pymeasure.instruments.kepco     module, 407  pymeasure.instruments.keysight     module, 408  pymeasure.instruments.kuhneelectronic     module, 441  pymeasure.instruments.lakeshore
pymeasure.experiment.parameters     module, 67  pymeasure.experiment.procedure     module, 65  pymeasure.experiment.results     module, 72  pymeasure.experiment.workers     module, 71  pymeasure.instruments     module, 95  pymeasure.instruments.activetechnologies     module, 111  pymeasure.instruments.aculight     module, 116  pymeasure.instruments.advantest     module, 117  pymeasure.instruments.advantest.advantestR37     module, 117  pymeasure.instruments.advantest.advantestR62     module, 139  pymeasure.instruments.agilent     module, 148  pymeasure.instruments.agilent.agilent4156     module, 165	pymeasure.instruments.eurotest     module, 253  pymeasure.instruments.fluke     module, 256  pymeasure.instruments.fwbell     module, 256  pymeasure.instruments.hcp     module, 260  pymeasure.instruments.heidenhain     module, 259  pymeasure.instruments.hp     module, 261  pymeasure.instruments.inficon     module, 319  pymeasure.instruments.ipgphotonics     module, 321  6769measure.instruments.keithley     module, 322  4Xpymeasure.instruments.kepco     module, 407  pymeasure.instruments.keysight     module, 408  pymeasure.instruments.kuhneelectronic     module, 441

module, 455	module, 625	
pymeasure.instruments.mksinst	pymeasure.instruments.thorlabs	
module, 470	module, 626	
<pre>pymeasure.instruments.newport</pre>	pymeasure.instruments.thyracont	
module, 476	module, 627	
pymeasure.instruments.ni	pymeasure.instruments.toptica	
module, 478	module, 633	
pymeasure.instruments.novanta	pymeasure.instruments.validators	
module, 491	module, 108	
pymeasure.instruments.oxfordinstruments module, 493	<pre>pymeasure.instruments.velleman module,635</pre>	
pymeasure.instruments.parker module, 505	pymeasure.instruments.yokogawa module,637	
pymeasure.instruments.pendulum	pymeasure.test	
module, 507	module, 57	
pymeasure.instruments.philips		
module, 506	Q	
<pre>pymeasure.instruments.proterial</pre>	QListener (class in pymeasure.display.listeners	s), 79
module, 508	<pre>query_ac_current()</pre>	(pymea-
<pre>pymeasure.instruments.ptw module, 511</pre>	sure.instruments.ni.virtualbench.Virtua method), 481	alBench.DigitalMultimete
pymeasure.instruments.racal	query_acquisition_status()	(pymea-
module, 518	sure.instruments.ni.virtualbench.Virtua	alBench.MixedSignalOsci
<pre>pymeasure.instruments.razorbill</pre>	method), 486	
module, 521	<pre>query_adc_setup()</pre>	(pymea-
pymeasure.instruments.redpitaya module, 522	sure.instruments.agilent.agilentB1500. method), 187	AgilentB1500
pymeasure.instruments.rigol	query_analog_channel()	(pymea-
module, 524	sure.instruments.ni.virtualbench.Virtua	alBench.MixedSignalOsci
pymeasure.instruments.rohdeschwarz	method), 486	
module, 525	query_analog_channel_characteristics(	)
pymeasure.instruments.santec	(pymeasure.instruments.ni.virtualbenc	h.VirtualBench.MixedSig
module, 544	method), 486	
pymeasure.instruments.siglenttechnologies	<pre>query_analog_edge_trigger()</pre>	(pymea-
module, 545 pymeasure.instruments.signalrecovery	sure.instruments.ni.virtualbench.Virtua	ılBench.MixedSignalOsci
module, 550	method), 486	(
pymeasure.instruments.spellmanhv	query_analog_pulse_width_trigger()	(pymea- alPanah MiyadSianalOsa)
module, 564	sure.instruments.ni.virtualbench.Virtua method), 487	ubench.MixeasignaiOsci
pymeasure.instruments.srs	query_arbitrary_waveform()	(рутеа-
module, 567	sure.instruments.ni.virtualbench.Virtua	* *
pymeasure.instruments.tcpowerconversion	method), 483	
module, 585	query_arbitrary_waveform_gain_and_off	set()
pymeasure.instruments.tdk	(pymeasure.instruments.ni.virtualbence	
module, 589	method), 483	
<pre>pymeasure.instruments.tektronix</pre>	<pre>query_current_output()</pre>	(pymea-
module, 598	sure.instruments.ni.virtualbench.Virtua	alBench.PowerSupply
pymeasure.instruments.teledyne	method), 489	
module, 599	<pre>query_dc_current()</pre>	(рутеа-
pymeasure.instruments.temptronic	sure.instruments.ni.virtualbench.Virtua	alBench.DigitalMultimete
module, 613	method), 481	,
pymeasure.instruments.texio	<pre>query_dc_voltage()</pre>	(pymea-
module, 622 pymeasure.instruments.thermotron	sure.instruments.ni.virtualbench.Virtua	uBench.DigitalMultimete
pymeasure. This cruments. thermotron	method), 481	

```
query_enabled_analog_channels()
                                            (pymea- query_series_resistor()
                                                                                                   (pymea-
        sure.instruments.ni.virtualbench.VirtualBench.MixedSignal@xcellinstrapments.agilent.agilentB1500.AgilentB1500
        method), 487
                                                               method), 189
query_event_status_register()
                                                      query_staircase_sweep_settings()
                                                                                                  (рутеа-
                                            (pymea-
        sure.instruments.anritsu.AnritsuMS464xB
                                                               sure.instruments.agilent.agilentB1500.AgilentB1500
        method), 241
                                                               method), 188
query_export_signal()
                                            (pymea- query_standard_waveform()
                                                                                                  (pymea-
        sure.instruments.ni.virtualbench.VirtualBench.DigitalInputQutpunstruments.ni.virtualbench.VirtualBench.FunctionGenerator
        method), 479
                                                               method), 483
                                            (pymea- query_time_stamp_setting()
query_generation_status()
                                                                                                  (рутеа-
        sure.instruments.ni.virtualbench.VirtualBench.FunctionGenenatoinstruments.agilent.agilentB1500.AgilentB1500
         method), 483
                                                               method), 187
query_learn()
                                            (pymea- query_timing()
                                                                                                  (pymea-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               sure.instruments.ni.virtualbench.VirtualBench.MixedSignalOscill
        method), 185
                                                               method), 487
query_learn()
                                            (pymea- query_trigger_delay()
                                                                                                   (рутеа-
        sure.instruments.agilent.agilentB1500.QueryLearn
                                                               sure.instruments.ni.virtualbench.VirtualBench.MixedSignalOscill
        static method), 192
                                                               method), 487
                                                     query_trigger_type()
query_learn()
                                            (pymea-
                                                                                                  (pymea-
        sure.instruments.agilent.agilentB1500.SMU
                                                               sure.instruments.ni.virtualbench.VirtualBench.MixedSignalOscill
        method), 190
                                                               method), 487
query_learn_header()
                                            (pymea- query_voltage_output()
                                                                                                  (рутеа-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               sure.instruments.ni.virtualbench.VirtualBench.PowerSupply
        method), 185
                                                               method), 489
                                            (pymea- query_waveform_mode()
query_learn_header()
                                                                                                   (pymea-
                                                               sure.instruments.ni.virtualbench.VirtualBench.FunctionGenerator
        sure.instruments.agilent.agilentB1500.QueryLearn
         class method), 192
                                                               method), 483
query_line_configuration()
                                            (pymea- QueryLearn
                                                                                                   рутеа-
                                                                           (class
                                                                                        in
        sure.instruments.ni.virtualbench.VirtualBench.DigitalInputQutpuinstruments.agilent.agilentB1500), 192
        method), 479
                                                      questionable_event_enabled
                                                                                                   (pymea-
query_meas_mode()
                                            (рутеа-
                                                               sure.instruments.keithley.Keithley6221
                                                                                                    prop-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               erty), 374
        method), 186
                                                      questionable_event_reg
                                                                                                  (pymea-
                                                               sure.instruments.rohdeschwarz.sfm.SFM
query_meas_op_mode()
                                            (рутеа-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               property), 533
        method), 189
                                                      questionable_events
                                                                                                  (pymea-
query_meas_range_current_auto()
                                            (pymea-
                                                               sure.instruments.keithley.Keithley6221
                                                                                                    prop-
        sure.instruments.agilent.agilentB1500.AgilentB1500
        method), 189
                                                      questionable_operation_enable_reg
                                                                                                  (рутеа-
query_meas_ranges()
                                                               sure.instruments.rohdeschwarz.sfm.SFM
                                            (pymea-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               property), 533
        method), 189
                                                      questionanble_status_reg
                                                                                                  (pymea-
                                                               sure.instruments.rohdeschwarz.sfm.SFM
query_meas_settings()
                                            (pymea-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               property), 533
                                                                   (pymeasure.display.manager.BaseManager
        method), 186
                                                      queue()
query_measurement()
                                            (рутеа-
                                                               method), 79
        sure.instruments.ni.virtualbench.VirtualBench.DigitalMaQihpytmeasure.display.windows.managed_window.ManagedWindowl
                                                               method), 93
        method), 481
query_modules()
                                            (pymea- queue_sequence()
                                                                                                   (рутеа-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               sure. display. widgets. sequencer\_widget. SequencerWidget
        method), 185
                                                               method), 87
                                            (pymea- quick_range()
                                                                           (pymeasure.instruments.srs.SR830
query_sampling_settings()
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               method), 578
        method), 188
```

R			su	re.instruments.tdk.tdk_gen40_38	.TDK_Gen40_38
R75 out	(pymeasure.instruments.rohdeschw	arz.sfm.SFM	m	ethod), 591	
<u></u>	property), 526	c. 2.5j51 1.1	ramp_to_c		(рутеа-
Racal19	92 (class in pymeasure.instruments	.racal), 519	SU	re.instruments.tdk.tdk_gen80_65	.TDK_Gen80_65
	measure.instruments.hp.hp856Xx.S		m	ethod), 596	
- 43	tribute), 304	1	ramp_to_c		(рутеа-
ramp()	(pymeasure.instruments.ami.AMI4.	30 method),		re.instruments.yokogawa.Yokoga	wa7651
	223		m	ethod), 638	
ramp_ra	te (pymeasure.instruments.tcpower	conversion.C	//ramp_to_f	ield() (pymeasure.instruments.	ami.AMI430
	property), 587		m	einoa), 223	
ramp_ra	te (pymeasure.instruments.temptro	nic.ATSBase	ramp_to_v	=	(рутеа-
	property), 617			re.instruments.keithley.Keithley2	400
ramp_ra	te_current	(pymea-		ethod), 348	,
	sure.instruments.ami.AMI430	property),	ramp_to_v		(pymea-
	223			re.instruments.keithley.Keithley2	430
ramp_ra	te_field (pymeasure.instruments.	ami.AMI430		ethod), 356	(manage of g
	property), 223		ramp_to_v		(pymea-
ramp_so		(pymea-		re.instruments.keithley.Keithley6	31/ <b>B</b>
	sure.instruments.agilent.agilentB1	500.SMU	ramp_to_v	ethod), 380	(pymag
	method), 191		_	=	(pymea- wa7651
ramp_st	art_power	(рутеа-		re.instruments.yokogawa.Yokoga ethod), 638	wa7031
	sure.instruments.tcpowerconversio	n.CXN	ramp_to_z		(рутеа-
	property), 587		-	ero() vre.instruments.deltaelektronika.S	* *
ramp_sy		(pymea-		ethod), 253	M/043D
	sure.instruments.agilent.Agilent33	220A	ramp_to_z		(рутеа-
	property), 173	,		re.instruments.eurotest.EurotestH	
ramp_sy	-	(pymea-		ethod), 255	11 1 120230
	sure.instruments.agilent.Agilent33.	500 prop-		neasure.instruments.fwbell.FWBe	115080 prop-
	erty), 177	,		ty), 258	usooo prop
ramp_sy	mmetry	(pymea-	caramae (nvi	measure instruments hn HP34374	( nronerty)
	sure.instruments.agilent.agilent33.	000.Agilenī33.	0 <del>00Erranne</del> r 26	57	r property),
<b>****</b>	property), 180	(**************************************		neasure.instruments.hp.HP3478A	A property).
ramp_sy		(pymea-		70	1 p. spe. 15),
	sure.instruments.keysight.Keysight	01100A		measure.instruments.hp.HP437B	property).
namn +i	property), 437	VC nuon autu)		13	F. F. Sylvi
ramp_ti	me (pymeasure.instruments.aja.DC2 219	AS property),	range (pym	easure.instruments.keithley.Keith	leyDMM6500
ramn +i	me (pymeasure.instruments.proteria	l rod PODA		roperty), 390	
ramp_tr	property), 511	1.1044.ROD4	JIIIIII -	easure.instruments.keithley.keithi	leyDMM6500.ScannerCard200
ramn to	_current()	(рутеа-		roperty), 394	
ramp_co	sure.instruments.ami.AMI430 meth		range (pyr	measure.instruments.lakeshore.La	akeShore425
ramn to	_current()	(pymea-	pi	roperty), 454	
ramp_co	sure.instruments.danfysik.Danfysik	* *	range (pym	easure.instruments.lakeshore.lak	eshore_base.LakeShoreHeater
	method), 250	.0500	рі	roperty), 455	
ramn to	_current()	(рутеа-	range (py	measure.instruments.ptw.ptwUNI	DOS prop-
ramp_co	sure.instruments.deltaelektronika.S		er	rty), 516	
	method), 252	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	range (pym	easure.instruments.thyracont.smc	artline_v2.SmartlineV2
ramp to	_current()	(рутеа-	pı	roperty), 631	
ramp_co	sure.instruments.keithley.Keithley2	* *	range_ (	oymeasure.instruments.hp.HP344	201A prop-
	method), 348		er	rty), 265	
ramp to	_current()	(рутеа-	range_max	**	ptwUNIDOS
	sure.instruments.keithley.Keithley2	* *	pı	roperty), 516	
	method), 355		range_res	(pymeasure.instruments.ptw.)	ptwUNIDOS
ramp_to	_current()	(рутеа-	pı	roperty), 516	
	The state of the s	* *			

Ranging (class in pymea- sure.instruments.agilent.agilentB1500), 193	read() (pymeasure.instruments.keithley.Keithley2600 method), 363
rate (pymeasure.instruments.inficon.sqm160.SensorChar property), 321	
ratio (pymeasure.instruments.agilent.agilent4156.VARD property), 170	
ratio (pymeasure.instruments.signalrecovery.DSP7225 property), 555	method), 374
ratio (pymeasure.instruments.signalrecovery.DSP7265 property), 561	read() (pymeasure.instruments.keithley.Keithley6517B method), 381
razorbillRP100 (class in pymea- sure.instruments.razorbill), 521	read() (pymeasure.instruments.keithley.KeithleyDAQ6510 method), 402
read() (pymeasure.adapters.Adapter method), 47 read() (pymeasure.adapters.FakeAdapter method), 59	read() (pymeasure.instruments.keysight.Keysight81160A method), 437
read() (pymeasure.adapters.PrologixAdapter method), 55	read() (pymeasure.instruments.keysight.KeysightE36312A method), 419
read() (pymeasure.adapters.SerialAdapter method), 52 read() (pymeasure.adapters.VISAAdapter method), 49	read() (pymeasure.instruments.keysight.KeysightE3631A method), 427
read() (pymeasure.instruments.aja.DCXS method), 219 read() (pymeasure.instruments.andeenhagerling.AH2700	read() (pymeasure.instruments.lecroy.LeCroyT3DSO1204 0A method), 464
method), 229 read() (pymeasure.instruments.attocube.anc300.ANC300	read() (pymeasure.instruments.mksinst.mksinst.MKSInstrument OController method), 471
method), 246 read() (pymeasure.instruments.Channel method), 106	read() (pymeasure.instruments.ni.virtualbench.VirtualBench.DigitalInput( method), 479
read() (pymeasure.instruments.danfysik.Danfysik8500 method), 250	read() (pymeasure.instruments.ni.virtualbench.VirtualBench.DigitalMultin method), 481
read() (pymeasure.instruments.fluke.Fluke7341 method), 256	read() (pymeasure.instruments.parker.ParkerGV6 method), 505
read() (pymeasure.instruments.fwbell.FWBell5080 method), 258	method), 507
read() (pymeasure.instruments.hcp.TC038 method), 261 read() (pymeasure.instruments.hcp.TC038D method),	read() (pymeasure.instruments.proterial.ROD4 method), 510
261 read() (pymeasure.instruments.inficon.sqm160.SQM160	read() (pymeasure.instruments.ptw.ptwDIAMENTOR method), 513
method), 320 read() (pymeasure.instruments.Instrument method), 104	read() (pymeasure.instruments.ptw.ptwUNIDOS method), 516
read() (pymeasure.instruments.ipgphotonics.yar.YAR method), 322	read() (pymeasure.instruments.racal.Racal1992 method), 520
read() (pymeasure.instruments.keithley.Keithley2000 method), 327	read() (pymeasure.instruments.signalrecovery.DSP7225 method), 555
read() (pymeasure.instruments.keithley.Keithley2182 method), 396	read() (pymeasure.instruments.signalrecovery.DSP7265 method), 561
read() (pymeasure.instruments.keithley.Keithley2260B method), 336	read() (pymeasure.instruments.spellmanhv.SpellmanXRV method), 566
read() (pymeasure.instruments.keithley.Keithley2281S method), 339	read() (pymeasure.instruments.tcpowerconversion.CXN method), 587
read() (pymeasure.instruments.keithley.Keithley2306 method), 342	read() (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38 method), 592
read() (pymeasure.instruments.keithley.Keithley2400 method), 348	read() (pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65 method), 596
read() (pymeasure.instruments.keithley.Keithley2450 method), 356	method), 600
read() (pymeasure.instruments.keithley.Keithley2510 method), 360	read() (pymeasure.instruments.texio.TexioPSW360L30 method), 623

read() (pymeasure.instruments.thyracont.smar. method), 628	tline_v1.Sn	nartlineV1 sure.instruments.keithley.Keithley2400 method), 348	
read() (pymeasure.instruments.thyracont.smar. method), 631	tline_v2.Sn	ก <b>aretide</b> ได้ใกละry_values() sure.instruments.keithley.Keithley2450	(рутеа-
read() (pymeasure.instruments.toptica.ibeamsr	nart.IBeam		
method), 634		read_binary_values()	(pymea-
read() (pymeasure.instruments.velleman.Velleman.	nanK8090	sure.instruments.keithley.Keithley2510	
method), 636		method), 360	
<pre>read_analog_digital_dataframe()</pre>	(рутеа-	read_binary_values()	(pymea-
sure.instruments.ni.virtualbench.Virtu	alBench.M	ixedSignal <b>Osvėliastapn</b> ents.keithley.Keithley2600	
method), 487		method), 363	
read_analog_digital_u64()	(pymea-	read_binary_values()	(pymea-
sure.instruments.ni.virtualbench.Virtu method), 488	alBench.M	ixedSignal <b>@uvėlinstrapu</b> ents.keithley.Keithley2700 method), 367	
read_binary_values() (pymeasure.adapter.	s.Adapter		(рутеа-
method), 47	I	sure.instruments.keithley.Keithley2750	43
read_binary_values()	(рутеа-	method), 369	
sure.adapters.FakeAdapter method), 5	4 .	read_binary_values()	(рутеа-
read_binary_values()	(рутеа-	sure.instruments.keithley.Keithley6221	4)
- · · · · · · · · · · · · · · · · · · ·	method),	method), 374	
56	,,	read_binary_values()	(pymea-
<pre>read_binary_values()</pre>	(рутеа-	sure.instruments.keithley.Keithley6517	4 .
sure.adapters.SerialAdapter method),		method), 381	
read_binary_values()		read_binary_values()	(pymea-
sure.adapters.VISAAdapter method), 5		sure.instruments.keithley.KeithleyDAQ	6510
read_binary_values()	(рутеа-	method), 403	
sure.instruments.andeenhagerling.AH	2700A	read_binary_values()	(pymea-
method), 229		sure.instruments.keysight.Keysight8116	50A
read_binary_values()	(pymea-	method), 437	
sure.instruments.Channel method), 10	06	read_binary_values()	(pymea-
<pre>read_binary_values()</pre>	(pymea-	sure.instruments.keysight.KeysightE36.	312A
sure.instruments.fwbell.FWBell5080	method),	method), 419	
258		read_binary_values()	(pymea-
read_binary_values()	(рутеа-	sure.instruments.keysight.KeysightE36	31A
sure.instruments.Instrument method),		method), 427	,
<pre>read_binary_values()</pre>		read_binary_values()	(pymea-
sure.instruments.keithley.Keithley2000	)	sure.instruments.lecroy.LeCroyT3DSO	1204
method), 327	,	method), 465	,
read_binary_values()	(pymea-	read_binary_values()	(pymea-
sure.instruments.keithley.Keithley2182 method), 396	2	sure.instruments.proterial.ROD4 510	method),
read_binary_values()	(pymea-	read_binary_values()	(pymea-
sure.instruments.keithley.Keithley2200 method), 333	)	sure.instruments.signalrecovery.DSP72 method), 555	225
<pre>read_binary_values()</pre>	(рутеа-	read_binary_values()	(pymea-
sure.instruments.keithley.Keithley2260	$\partial B$	sure.instruments.signalrecovery.DSP72	265
method), 336		method), 561	
read_binary_values()	(pymea-	read_binary_values()	(pymea-
sure.instruments.keithley.Keithley2281 method), 339	1S	sure.instruments.tdk.tdk_gen40_38.TD method), 592	K_Gen40_38
read_binary_values()	(рутеа-	read_binary_values()	(рутеа-
sure.instruments.keithley.Keithley2300 method), 342		sure.instruments.tdk.tdk_gen80_65.TD method), 596	
read_binary_values()	(рутеа-	read_binary_values()	(рутеа-
- · · · · · · ·	A >		4 2

sure. instruments. teledyne. Teledyne T3AF0	G	method), 360	
method), 601			(рутеа-
read_binary_values() (p sure.instruments.texio.TexioPSW360L30	рутеа-	sure.instruments.keithley.Keithley2600 method), 363	
method), 624			(рутеа-
	утеа-	sure.instruments.keithley.Keithley2700	4.7
sure.instruments.pendulum.cnt91.CNT91		method), 367	
method), 508		read_bytes()	(рутеа-
read_bytes() (pymeasure.adapters.Adapter me	ethod),	sure.instruments.keithley.Keithley2750 method), 370	
read_bytes() (pymeasure.adapters.FakeA	dapter	read_bytes()	(рутеа-
method), 59		sure.instruments.keithley.Keithley6221	
<pre>read_bytes() (pymeasure.adapters.PrologixA</pre>		method), 375	
method), 56		-	(рутеа-
read_bytes() (pymeasure.adapters.SerialA method), 52		sure.instruments.keithley.Keithley65171 method), 381	3
read_bytes() (pymeasure.adapters.VISAA	dapter		(рутеа-
method), 50		sure.instruments.keithley.KeithleyDAQ6	5510
, u	oymea-	method), 403	,
sure.instruments.andeenhagerling.AH270	OOA	•	(pymea-
method), 229	la arra al	sure.instruments.keysight.Keysight8116	OOA
read_bytes() (pymeasure.instruments.Cl method), 106		<pre>method), 437 read_bytes()</pre>	(рутеа-
	оутеа-	sure.instruments.keysight.KeysightE365	
sure.instruments.fwbell.FWBell5080 me		method), 419	,1211
258			(рутеа-
read_bytes() (pymeasure.instruments.Instr method), 104		sure.instruments.keysight.KeysightE365 method), 427	* *
	утеа-		(рутеа-
sure.instruments.keithley.Keithley2000		sure.instruments.lecroy.LeCroyT3DSO	1204
method), 327		method), 465	
read_bytes() (p	рутеа-	<pre>read_bytes() (pymeasure.instruments.proteria</pre>	ıl.ROD4
sure.instruments.keithley.Keithley2182		method), 510	
method), 396		•	(рутеа-
	рутеа-	sure.instruments.signalrecovery.DSP72	25
sure.instruments.keithley.Keithley2200		method), 555	,
method), 333			(pymea-
	оутеа-	sure.instruments.signalrecovery.DSP72	03
sure.instruments.keithley.Keithley2260B method), 336		<pre>method), 561 read_bytes()</pre>	(рутеа-
	оутеа-	sure.instruments.tdk.tdk_gen40_38.TDl	
sure.instruments.keithley.Keithley2281S	ymea	method), 592	1_001110_50
method), 339			(рутеа-
	утеа-	sure.instruments.tdk.tdk_gen80_65.TDI	
sure.instruments.keithley.Keithley2306		method), 596	
method), 342		read_bytes()	(рутеа-
read_bytes() (p	оутеа-	sure.instruments.teledyne.TeledyneT3A	FG
sure.instruments.keithley.Keithley2400		method), 601	
method), 348		read_bytes()	(рутеа-
*	рутеа-	sure.instruments.texio.TexioPSW360L3	0
sure.instruments.keithley.Keithley2450		method), 624	(
method), 356			(pymea- NailantP1500
read_bytes() (p sure.instruments.keithley.Keithley2510	рутеа-	sure.instruments.agilent.agilentB1500.1 method), 189	agueni <b>B</b> 1300

read_data() (pymeasure.instruments.agilent.a method), 189	agilentB150	OO <b>c&amp;gildrltBdb00</b> () sure.instruments.hp.hp856Xx.HP856X	(pymea-
read_data() (pymeasure.instruments.hp.	HP3437A	method), 295	
method), 267	,	recall_trace()	(рутеа-
read_datafile()	(pymea-	sure.instruments.hp.hp856Xx.HP856X	x
sure.instruments.anritsu.AnritsuMS46	04XB	method), 287	Listonon
method), 241	(	receive() (pymeasure.experiment.listeners method), 64	s.Listener
read_detector()	(pymea-		(ang) 61
sure.instruments.ptw.ptwUNIDOS 516	method),	Recorder (class in pymeasure.experiment.listen RedPitayaScpi (class in	pymea-
read_measurement()	(рутеа-	sure.instruments.redpitaya.redpitaya_s	1 2
sure.instruments.advantest.advantestk			scpi),
method), 127	10271.7141	reference(pymeasure.instruments.signalrecov	ery DSP7225
read_measurement()	(рутеа-	property), 555	cry.D51 7225
sure.instruments.philips.PM6669		reference (pymeasure.instruments.signalrecov	ery DSP7265
507		property), 562	cry.D51 7205
read_measurement_from_addr()			(рутеа-
	R624X.SMU	JChannel sure.instruments.srs.SR860 property),	
method), 137		reference_level	(рутеа-
read_memory()	(рутеа-	sure.instruments.hp.hp856Xx.HP856X	x at-
sure.instruments.anritsu.AnritsuMS97	710C	tribute), 278	
method), 232	,	reference_level	(pymea-
read_output()	(рутеа-	sure.instruments.yokogawa.aq6370ser	ies.AQ6370Serie
sure.instruments.ni.virtualbench.Virtu	ialBench.Pe		,
method), 489	,	reference_level_calibration	(рутеа-
read_random_memory()	(pymea-		x at-
sure.instruments.advantest.advantestR	(024X.SM C		(
method), 136	(	reference_offset	(pymea-
read_trace()	(pymea-	sure.instruments.hp.hp856Xx.HP856X	x at-
sure.instruments.rohdeschwarz.fsserie	es.FSL	tribute), 278	(
method), 541	(	reference_output	(pymea-
read_trace()	(pymea-	sure.instruments.anapico.APSIN12G p 227	property),
sure.instruments.rohdeschwarz.fsserie method), 542	S.F S W	reference_phase	(рутеа-
readAI() (in module pymeasure.instruments	comadi)	sure.instruments.signalrecovery.DSP7.	4.7
110		property), 555	223
reading (pymeasure.instruments.hp.HP3440)	lA prop-		(pymea-
erty), 265		sure.instruments.signalrecovery.DSP7	265
reading_available	(рутеа-	property), 562	
sure.instruments.keithley.Keithley228. property), 339	1S	reference_source (pymeasure.instruments.s property), 578	rs.SR830
recall() (pymeasure.instruments.tdk.tdk_gen4	40_38.TDK	_ <b>&amp;febeloen8e_source</b> (pymeasure.instruments.s	rs.SR860
method), 592	00 65 TDV	property), 582	(
recall() (pymeasure.instruments.tdk.tdk_gen8 method), 596	00_03.1 <i>D</i> K	sure.instruments.srs.SR830 property),	(pymea- 578
recall_config()	(nymaa	reference_source_trigger	(pymea-
=		base.SPDBa <b>ss</b> ee.instruments.srs.SR860 property),	4 -
method), 545	igieni_spai	reference_triggermode	(pymea-
recall_open_short_average()	(рутеа-	sure.instruments.srs.SR860 property),	* *
sure.instruments.hp.hp856Xx.HP856X		reflection_limit	(pymea-
method), 294		sure.instruments.kuhneelectronic.Kusg	
recall_state()	(рутеа-	property), 443	,
sure.instruments.hp.hp856Xx.HP856X		refresh_parameters()	(рутеа-
method), 277		sure.experiment.procedure.Procedure	

	1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1
66	attribute), 475
regulation_mode (pymeasure.instruments.aja.DCXS property), 219	relay_4 (pymeasure.instruments.mksinst.mks937b.MKS937B attribute), 473
	500 ay_5 (pymeasure.instruments.mksinst.mks937b.MKS937B
property), 390	attribute), 473
* *	relay_6 (pymeasure.instruments.mksinst.mks937b.MKS937B
sure.instruments.keithley.KeithleyDMM6500	attribute), 473
property), 390	relay_7 (pymeasure.instruments.mksinst.mks937b.MKS937B
relative_field (pymea-	attribute), 473
sure.instruments.lakeshore.LakeShore421 property), 452	relay_8 (pymeasure.instruments.mksinst.mks937b.MKS937B attribute), 473
relative_field_raw (pymea-	relay_9 (pymeasure.instruments.mksinst.mks937b.MKS937B
sure.instruments.lakeshore.LakeShore421	attribute), 473
property), 452	relay_mode (pymeasure.instruments.advantest.advantestR624X.SMUChan
relative_mode_enabled (pymea-	property), 138
sure.instruments.hp.HP437B property), 314	RelayChannel (class in pymea-
relative_mode_enabled (pymea-	sure.instruments.mksinst.mksinst), 471
sure.instruments.lakeshore.LakeShore421	release_control() (pymea-
property), 453	sure.instruments.tcpowerconversion.CXN
relative_multiplier (pymea-	method), 587
sure.instruments.lake shore.Lake Shore 421	reload() (pymeasure.experiment.results.Results
property), 453	method), 73
relative_setpoint (pymea-	remaining_deposition_time_min (pymea-
sure.instruments.lakeshore.LakeShore421	sure.instruments.aja.DCXS property), 219
property), 453	remaining_deposition_time_sec (pymea-
relative_setpoint_multiplier (pymea-	sure.instruments.aja.DCXS property), 219
sure.instruments.lakeshore.LakeShore421	remote (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38
property), 453	property), 592
	remote(pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65
sure.instruments.lakeshore.LakeShore421	property), 596
property), 453	remote() (pymeasure.instruments.danfysik.Danfysik8500
	- · · · · · · · · · · · · · · · · · · ·
Relay (class in pymea-	method), 250
sure.instruments.mksinst.mks937b), 474	remote() (pymeasure.instruments.keithley.Keithley2000
Relay (class in pymea-	
	remote_control_enabled (pymea-
relay() (pymeasure.instruments.keithley.Keithley2306	sure.instruments.hp.HP34401A property),
method), 342	265
relay_1 (pymeasure.instruments.mksinst.mks937b.MKS93	Memote_interfaces (pymea-
attribute), 473	sure.instruments.rohdeschwarz.sfm.SFM
relay_1 (pymeasure.instruments.mksinst.mks974b.MKS97	74B property), 533
attribute), 475	remote_local_state (pymea-
relay_10 (pymeasure.instruments.mksinst.mks937b.MKS9	• • • • • • • • • • • • • • • • • • • •
attribute), 473	property), 173
relay_11 (pymeasure.instruments.mksinst.mks937b.MKS9	
attribute), 473	sure.instruments.keithley.Keithley2000
relay_12 (pymeasure.instruments.mksinst.mks937b.MKS9	
attribute), 473	**
relay_2 (pymeasure.instruments.mksinst.mks937b.MKS93	
attribute), 473	265
	Affemote_mode (pymeasure.instruments.temptronic.ATSBase
attribute), 475	property), 617
relay_3 (pymeasure.instruments.mksinst.mks937b.MKS93	Themote_trigger_type (pymea-
attribute), 473	sure.instruments.anritsu.AnritsuMS464xB
relay_3 (pymeasure.instruments.mksinst.mks974b.MKS97	74B property), 241

remove() (pymeasure.display.manager.Base method), 80	Manager	replace	_placeholders() (in module pymea- sure.experiment.results), 73
remove() (pymeasure.display.manager.	Manager	request	
method), 80		1	sure.instruments.tcpowerconversion.CXN
<pre>remove() (pymeasure.display.widgets.image_w</pre>	idget.Image	eWidget	method), 587
method), 84		request	_service() (pymea-
$\verb"remove()" (pymeasure.display.widgets.plot\_widgets.plot") and the substitution of t$	get.PlotWic	lget	sure.instruments.hp.hp856Xx.HP856Xx
method), 85			method), 276
remove() (pymeasure.display.widgets.tab_widg	et.TabWidg	g <b>a</b> request	
method), 88	J4 T.,1.1 .1	I7: J4	sure.instruments.hp.hp856Xx.HP856Xx at-
remove() (pymeasure.display.widgets.table_widenthod), 91	aget.1abiev	<i>viagei</i> res_ban	tribute), 277 dwidth (pymea-
remove_child()	(рутеа-	165_Dan	sure.instruments.rohdeschwarz.fsseries.FSSeries
sure.instruments.common_base.Comn			property), 540
method), 102	.0.120.50	reset()	(pymeasure.adapters.PrologixAdapter method),
remove_child()	(рутеа-	•	56
sure.instruments.keithley.Keithley2200		reset()	(pymeasure.instruments.activetechnologies.AWG401x_AWG.Wave
method), 333			method), 112
remove_child()	(pymea-	reset()	(pymeasure.instruments.agilent.agilentB1500.AgilentB1500
sure.instruments.keysight.Keysight811	60A		method), 185
method), 437	,	reset()	(pymeasure.instruments.andeenhagerling.AH2700A
remove_child()	(pymea-		method), 229
sure.instruments.keysight.KeysightE36 method), 419	0312A	reset()	(pymeasure.instruments.fwbell.FWBell5080 method), 258
remove_child()	(рутеа-	reset()	(pymeasure.instruments.generic_types.SCPIMixin
sure.instruments.keysight.KeysightE36		16366()	method), 108
method), 427		reset()	(pymeasure.instruments.hp.HP437B method),
remove_child()	(рутеа-		314
sure.instruments.lecroy.LeCroyT3DSC	01204	reset()	(pymeasure.instruments.hp.HP8116A method),
method), 465			273
remove_file()			(pymeasure.instruments.hp.HP8657B method),
sure.instruments.activetechnologies.A	WG401x_A		307
method), 113	,	reset()	(pymeasure.instruments.hp.HP8753E method),
remove_node()	(pymea-	"Food4 (A	308
method), 87			e(pymeasure.instruments.hp.HPLegacyInstrument method), 316
remove_unit_suffix()			(pymeasure.instruments.Instrument method),
sure.instruments.aimtti.ld400p.LD400 method), 218	P class		(pymeasure.instruments.keithley.Keithley2000
rename_channel()	(рутеа-	reset()	method), 327
sure.instruments.rohdeschwarz.fsserie		reset()	(pymeasure.instruments.keithley.Keithley2182
method), 542		()	method), 397
repeat (pymeasure.instruments.tdk.tdk_gen40_	38.TDK_G	ene(8 <u>e</u> 18)	
property), 592			method), 333
$\verb"repeat" (pymeasure.instruments.tdk.tdk\_gen80\_$	65.TDK_G	en80 <u>e</u> 6.()	(pymeasure.instruments.keithley.Keithley2260B
property), 597			method), 336
repeat_sweep()	(pymea-	reset()	(pymeasure.instruments.keithley.Keithley2281S
sure.instruments.anritsu.AnritsuMS97	40A		method), 339
<pre>method), 233 repetition_rate</pre>	(рутеа-	reset()	(pymeasure.instruments.keithley.Keithley2306 method), 343
	(pymea- property),	reset()	
273	operty),	10000	method), 348
repetitions (pymeasure.instruments.rohdesch	warz.hmp.	HM484040	
property), 543	•		method), 356

reset()	(pymeasure.instruments.keithley.Keithley2510 method), 361	<pre>method), 397 reset_buffer()</pre>	(рутеа-
reset()	(pymeasure.instruments.keithley.Keithley2600 method), 364	sure.instruments.keithley.Keithley2400 method), 348	* *
reset()	(pymeasure.instruments.keithley.Keithley2700 method), 367	reset_buffer()  sure.instruments.keithley.Keithley2450	(pymea-
reset()	(pymeasure.instruments.keithley.Keithley2750 method), 370	method), 356 reset_buffer()	
reset()	(pymeasure.instruments.keithley.Keithley6221 method), 375	sure.instruments.keithley.Keithley2700 method), 367	(pymea- )
reset()	(pymeasure.instruments.keithley.Keithley6517B method), 381		(рутеа-
reset()	(pymeasure.instruments.keithley.KeithleyDAQ651 method), 403	· · · · · · · · · · · · · · · · · · ·	(pymea-
reset()	(pymeasure.instruments.keysight.Keysight81160A method), 437		• •
reset()	(pymeasure.instruments.keysight.KeysightE36312 method), 419		(pymea- 06510
reset()	(pymeasure.instruments.keysight.KeysightE3631A method), 427		
reset()	(pymeasure.instruments.lecroy.LeCroyT3DSO120 method), 465		(pymea-
reset()		sure.instruments.ni.virtualbench.Virtualbench, 479	4.5
reset()			(pymea- alBench.DivitalMultimete
reset()	(pymeasure.instruments.ptw.ptwDIAMENTOR method), 513	<pre>method), 481 reset_instrument()</pre>	(рутеа-
reset()		sure.instruments.ni.virtualbench.Virtu method), 484	4.7
reset()	(pymeasure.instruments.signalrecovery.DSP7225 method), 555		(pymea- alBench.MixedSignalOsci
reset()	(pymeasure.instruments.signalrecovery.DSP7265 method), 562		(рутеа-
reset()		sure.instruments.ni.virtualbench.Virtua method), 489	* *
reset()	(pymeasure.instruments.tdk.tdk_gen40_38.TDK_0 method), 592		(pymea- 00
reset()	) (pymeasure.instruments.tdk.tdk_gen80_65.TDK_0		
	method), 597	reset_measurement()	(рутеа-
reset()	(pymeasure.instruments.teledyne.TeledyneT3AFG method), 601	sure.instruments.racal.Racal1992 520	method),
reset()	(pymeasure.instruments.temptronic.ATSBase method), 617	reset_OVP_OCP() sure.instruments.hp.HP6632A method	( <i>pymea</i> - ), 318
reset()	(pymeasure.instruments.texio.TexioPSW360L30 method), 624	reset_position() sure.instruments.anaheimautomation.l	(pymea- DPSeriesMotorController
reset_a	alarm() (pymea-	method), 226	
	sure. instruments. lake shore. Lake Shore 211	reset_settings()	(рутеа-
_	method), 446	sure.instruments.agilent.agilent4156.A	agilentMeasurementChani
reset_b	ouffer() (pymea-	method), 169	/
	sure.instruments.keithley.Keithley2000	reset_system_parameters()	(pymea-
roco+ L	method), 327 ouffer() (pymea-	sure.instruments.inficon.sqm160.SQM	100
TESET_D	ouffer() (pymea- sure.instruments.keithley.Keithley2182	<pre>method), 321 reset_thickness_rate()</pre>	(рутеа-

sure.instruments.inficon.sqm160.SQM160		property), 390	
method), 321	resis	stance_4w_range	(pymea-
**	утеа-	sure.instruments.agilent.Agilent344.	50A
sure.instruments.inficon.sqm160.SQM160		property), 164	
method), 321	resis	stance_4W_range	(рутеа-
reset_to_defaults() (py	утеа-	sure.instruments.keithley.Keithley20	000 prop-
	thod),	erty), 328	
507		stance_4W_range	(рутеа-
resetpoint (pymeasure.instruments.mksinst.mksin property), 471	st.RelayChann	nel sure.instruments.keithley.KeithleyD1 property), 390	MM6500
$\verb"resistance" (pymeasure. instruments. a gilent. A gilenter a gi$	<i>t34410A</i> resis	tance_4W_reference	(pymea-
property), 161		sure.instruments.keithley.Keithley20	000 prop-
$\verb"resistance" (pymeasure. instruments. a gilent. A gilenter a gi$	t34450A	erty), 328	
property), 164	resis	stance_4W_relative	(pymea-
resistance (pymeasure.instruments.agilent.Agilen property), 209	tB2985	sure.instruments.keithley.KeithleyDi property), 390	MM6500
	401A resis	stance_4W_relative_enabled	(рутеа-
property), 265		sure.instruments.keithley.KeithleyDi	MM6500
${\tt resistance}\ (pymeasure.instruments.keithley.Keithle$	ley2000	property), 390	
property), 327	resis	stance_4w_resolution	(pymea-
resistance (pymeasure.instruments.keithley.Keithley.roperty), 348	ley2400	sure.instruments.agilent.Agilent344. property), 164	50A
resistance(pymeasure.instruments.keithley.Keithl	<i>ley2450</i> resis		(рутеа-
property), 356		sure.instruments.agilent.Agilent344.	
resistance (pymeasure.instruments.keithley.Keithl		property), 164	
property), 381		stance_digits	(pymea-
resistance (pymeasure.instruments.keithley.Keithl	leyDAQ6510	sure.instruments.keithley.Keithley20	00 prop-
property), 403	I D.M.M.6500	erty), 328	(200000000
resistance (pymeasure.instruments.keithley.Keithl	ieyDNIN <b>iides</b> ous		(pymea-
property), 390		sure.instruments.keithley.KeithleyDi	WW0300
resistance (pymeasure.instruments.oxfordinstrum		c.npmaperly), 390 stance_high_limit	(mm a a
property), 503 resistance_4w (p)		sure.instruments.srs.ldc500series.L1	(pymea-
sure.instruments.agilent.Agilent34410A	утеа-		CooseriesTEC
property), 161	rocic	<pre>property), 571 stance_limits</pre>	(pymea-
	ymea-	sure.instruments.srs.ldc500series.Ll	
sure.instruments.agilent.Agilent34450A	тса	property), 571	ocoobenesi Ec
property), 164	resis	stance_low_limit	(рутеа-
resistance_4w (pymeasure.instruments.hp.HP34		sure.instruments.srs.ldc500series.Ll	
property), 265	70111	property), 571	200000000000000000000000000000000000000
	<i>mea</i> - resis	stance_nplc	(рутеа-
sure.instruments.agilent.Agilent34450A	,,,,,,,,	sure.instruments.keithley.Keithley20	* *
property), 164		erty), 328	1 1
1 1 1	<i>mea</i> - resis	tance_nplc	(рутеа-
	prop-	sure.instruments.keithley.Keithley24 erty), 348	* *
• •	<i>mea-</i> resis	stance_nplc	(рутеа-
sure.instruments.keithley.KeithleyDMM65		sure.instruments.keithley.Keithley24	* *
property), 390		erty), 356	
		stance_nplc	(рутеа-
sure.instruments.keithley.Keithley2000 erty), 328	prop-	sure.instruments.keithley.Keithley65 property), 381	17B
• •	<i>mea</i> - resis	stance_nplc	(рутеа-
sure instruments keithley KeithleyDMM65		sure instruments keithley KeithleyDe	

property), 403	property), 520
	resolution_actual (pymea-
sure.instruments.keithley.KeithleyDMM6500	sure.instruments.anritsu.AnritsuMS9710C
property), 390	property), 232
resistance_range (pymea-	resolution_bandwidth (pymea-
sure.instruments.agilent.Agilent34450A	sure.instruments.hp.hp856Xx.HP856Xx at-
property), 164	tribute), 284
-	resolution_bandwidth (pymea-
sure.instruments.agilent.AgilentB2985 property), 209	sure.instruments.yokogawa.aq6370series.AQ6370Series property), 641
	resolution_bandwidth_to_span_ratio (pymea-
sure.instruments.keithley.Keithley2000 prop-	sure.instruments.hp.hp856Xx.HP856Xx at-
erty), 328	tribute), 284
•	resolution_vbw (pymea-
sure.instruments.keithley.Keithley2400 prop-	sure.instruments.anritsu.AnritsuMS9710C
erty), 349	property), 232
resistance_range (pymea-	resolution_vbw (pymea-
sure.instruments.keithley.Keithley2450 prop-	sure.instruments.anritsu.AnritsuMS9740A
erty), 356	property), 233
resistance_range (pymea-	responsivity (pymea-
sure.instruments.keithley.Keithley6517B	sure. instruments. srs. ldc 500 series. LDC 500 Series PD
property), 381	property), 570
- ·	restart_averaging() (pymea-
sure.instruments.keithley.KeithleyDAQ6510	sure.instruments.agilent.agilentE5062A.VNAChannel
property), 403	method), 159
resistance_range (pymea-	
sure.instruments.keithley.KeithleyDMM6500	ResultsClass (pymea-
property), 390	sure.display.widgets.image_frame.ImageFrame
resistance_reference (pymea-	attribute), 83
sure.instruments.keithley.Keithley2000 prop-	- · · · · · · · · · · · · · · · · · · ·
erty), 328	sure.display.widgets.plot_frame.PlotFrame
resistance_relative (pymea- sure.instruments.keithley.KeithleyDMM6500	attribute), 84
property), 390	ResultsCurve (class in pymeasure.display.curves), 76 ResultsDialog (class in pymea-
resistance_relative_enabled (pymea-	ResultsDialog (class in pymea- sure.display.widgets.results_dialog), 85
sure.instruments.keithley.KeithleyDMM6500	ResultsImage (class in pymeasure.display.curves), 77
property), 390	ResultsTable (class in pymea-
resistance_resolution (pymea-	sure.display.widgets.table_widget), 90
sure.instruments.agilent.Agilent34450A	resume() (pymeasure.display.manager.BaseManager
property), 164	method), 80
	return_to_local() (pymea-
sure.instruments.srs.ldc500series.LDC500Series	sTEC sure.instruments.anritsu.AnritsuMS464xB
property), 571	method), 241
${\tt resolution} \ (py measure. instruments. an rits u. An rits uMS97 and the sum of the$	7/ <b>DE</b> veal_in_file_explorer() (pymea-
property), 232	$sure. display. windows. managed\_window. Managed Window Base$
${\tt resolution} ({\it pymeasure.instruments.anritsu.AnritsuMS97}) \\$	
property), 233	reverse_power_limit (pymea-
resolution (pymeasure.instruments.hp.HP34401A	sure.instruments.tcpowerconversion.CXN
property), 265	property), 587
resolution (pymeasure.instruments.hp.HP3478A prop-	
erty), 270	property), 443
	rf_enabled (pymeasure.instruments.tcpowerconversion.CXN
erty), 314 resolution (pymeasure.instruments.racal.Racal1992	property), 587
1 COOLUCTOR (pymeusure.instruments.racat.Kacat1992	rf_out_enabled (pymea-

sure.instruments.rohdeschwarz.sfm.SFM property), 534		tus (pymeas property), 1		s.activetech	nologies.AWG401x_AWG
rf_sweep_center (pymea-	C				
sure.instruments.rohdeschwarz.sfm.SFM	S				
property), 534	Sample (	pymeasure.ii	istruments.hp.h	p856Xx.De	etectionModes
rf_sweep_span (pymea-		attribute), 3	02		
sure.instruments.rohdeschwarz.sfm.SFM	sample_	continuous	sly()		(рутеа-
property), 534		sure.instrun	nents.keithley.K	eithley2182	2
rf_sweep_start (pymea-		method), 39	7		
sure.instruments.rohdeschwarz.sfm.SFM	sample_	continuous	sly()		(рутеа-
property), 534		sure.instrun	nents.keithley. <mark>K</mark>	eithley2400	)
rf_sweep_step (pymea-		method), 34	.9		
sure.instruments.rohdeschwarz.sfm.SFM	sample_	count (py	measure.instru	ments.hp.H	IP34401A
property), 534		property), 2	65	_	
rf_sweep_stop (pymea-			_strategy		(pymea-
sure.instruments.rohdeschwarz.sfm.SFM	-		nents.activetech	nologies.A	
property), 534		property), 1	13		
right_limit(pymeasure.instruments.newport.esp300.Axi	sample_:	frequency	(pymeasure.in	struments.s	srs.SR830
property), 477		property), 5			
ROD4 (class in pymeasure.instruments.proterial), 509		hold_mode			(pymea-
ROD4Channel (class in pymea-			nents.advantest	advantestR	2624X.SMUChannel
sure.instruments.proterial.rod4), 511		property), 1			
rom_version (pymeasure.instruments.hp.HP6632A			_strategy		(рутеа-
property), 318	_		nents.activetech	mologies Al	4.
round_up() (pymeasure.display.curves.ResultsImage		property), 1		noiogics.71	110401x_11110
method), 77	sample_n		14		(рутеа-
rowCount() (pymeasure.display.widgets.sequencer_widget	Sampie_1 Sequence	rTreeModel	nents vokogawa	. aa6370ser	ries 106370Series
method), 87		property), 6		.uq0570se1	ies.AQ03705eries
rowCount()(pymeasure.display.widgets.table_widget.Pana	d <b>asModeli</b> l	groperiy), o Base	(class	in	рутеа-
method), 89	Samprem		ients.advantest		1 *
RQS (pymeasure.instruments.hp.hp856Xx.StatusRegister		139	ienis.aavaniesi	.uuvuniesiN	.024A),
attribute), 303	CamplaM		(alass	in	mum a a
rsd (pymeasure.instruments.deltaelektronika.SM7045D	SampleMo		(class 1ents.advantest	in advantant	pymea-
property), 253		139	ienis.aavaniesi	.aavaniesi <b>r</b>	.024A),
run() (pymeasure.display.listeners.Monitor method), 79					(
run() (pymeasure.display.plotter.Plotter method), 81		_harmonic_		V., 11D056V	(pymea-
run() (pymeasure.display.widgets.estimator_widget.Estimat	ntorThread	sure.insirun	nents.hp.hp856.	<i>ах.п</i> Розо <i>а</i>	Cx at-
method), 82				- :14 :1 -	D1500 M M - J -
run() (pymeasure.experiment.workers.Worker method),				диепт.адие	ntB1500.MeasMode
71		attribute), 1			(
run() (pymeasure.instruments.keysight.KeysightDSOX110.	$c_{C}$	g_auto_abo		" .D1500	(pymea-
method), 409			nents.agilent.ag	gilentB1500	AgilentB1500
run() (pymeasure.instruments.lecroy.LeCroyT3DSO1204		method), 18			
method), 465	samplin	g_frequenc	-		(pymea-
run() (pymeasure.instruments.ni.virtualbench.VirtualBenc	h Function	sure.instrun	nents.hp.hp856	<i>Xx.HP</i> 856 <i>X</i>	Cx at-
method), 484		//	2		,
memou), +0+	sampling	g_mode analOscillos	aona u	D. 500	(pymea-
run() (pymeasure.instruments.ni.virtualbench.VirtualBenc method), 488				gilentB1500	0.AgilentB1500
	na - :	property), 1	88		
run() (pymeasure.instruments.teledyne.TeledyneOscillosco	usampling				(рутеа-
method), 604	0		nents.anritsu.Ai	nritsuMS97	10C
run() (pymeasure.instruments.thermotron.Thermotron3800		property), 2	32		
method), 625	sampling	g_points			(рутеа-
run_mode (pymeasure.instruments.activetechnologies.AWG				nritsuMS97	740A
property), 113		property), 2	33		

```
method), 441
sampling_rate
                                             (pymea-
        sure.instruments.activetechnologies.AWG401x AWGale (pymeasure.instruments.teledyne.teledyne oscilloscope.TeledyneOsc
        property), 114
                                                               property), 609
sampling_rate_max
                                            (pymea- scale_volt(pymeasure.instruments.rohdeschwarz.sfm.SFM
        sure.instruments.activetechnologies.AWG401x_AWG
                                                               property), 534
                                                      scaling(pymeasure.instruments.spellmanhv.SpellmanXRV
        property), 114
sampling_rate_min
                                            (pymea-
                                                               property), 566
        sure.instruments.active technologies.AWG401x\_AWsGan()
                                                               (pymeasure.instruments.agilent.Agilent8722ES
        property), 114
                                                               method), 152
                                                               (pymeasure.instruments.hp.HP8753E method),
sampling_source()
                                            (рутеа-
                                                      scan()
        sure.instruments.agilent.agilentB1500.SMU
                                                               308
                                                      scan_card_vmax
        method), 192
                                                                                                   (pymea-
                                                               sure.instruments.keithley.KeithleyDMM6500
sampling_timing()
                                            (pymea-
        sure.instruments.agilent.agilentB1500.AgilentB1500
                                                               property), 391
                                                      scan_channels
        method), 188
                                                                                                   (рутеа-
SamplingMode
                      (class
                                   in
                                             рутеа-
                                                               sure.instruments.keithley.KeithleyDMM6500
        sure.instruments.agilent.agilentB1500), 196
                                                               property), 391
SamplingPostOutput
                           (class
                                                      scan_channels_list
                                                                                                   (pymea-
                                             рутеа-
        sure.instruments.agilent.agilentB1500), 196
                                                               sure.instruments.keithley.KeithleyDMM6500
save() (pymeasure.instruments.agilent.agilent4156.Agilent4156
                                                               property), 391
                                                      scan_continuous()
        method), 168
                                                                                                   (pymea-
save() (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38 sure.instruments.agilent.Agilent8722ES
         method), 592
                                                               method), 152
save() (pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen8065ount (pymeasure.instruments.keithley.KeithleyDMM6500
        method), 597
                                                               property), 391
save_config()
                                            (pymea- scan_id (pymeasure.instruments.keithley.KeithleyDMM6500
         sure.instruments.siglenttechnologies.siglent_spdbase.SPDBaseoperty), 391
        method), 545
                                                      scan_interval
                                                                                                   (рутеа-
                                                               sure.instruments.keithley.KeithleyDMM6500
save_dock_layout()
                                            (рутеа-
        sure.display.widgets.dock_widget.DockWidget
                                                               property), 391
        method), 88
                                                      scan_iscomplete
                                                                                                   (pymea-
save_experiment_copy()
                                             (рутеа-
                                                               sure.instruments.keithley.KeithleyDMM6500
        sure.display.windows.managed_window.ManagedWindowBaseoperty), 391
                                                      scan_modes(pymeasure.instruments.keithley.KeithleyDMM6500
        method), 94
save_file() (pymeasure.instruments.activetechnologies.AWG401x \( \textbf{AWG}erty \), 391
        method), 114
                                                      scan_points(pymeasure.instruments.agilent.Agilent8722ES
save_graphics()
                                             (pymea-
                                                               property), 152
        sure.instruments.agilent.Agilent4294A
                                                      scan_points(pymeasure.instruments.agilent.agilentE5062A.VNAChannel
        method), 172
                                                               property), 159
save_sequence()
                                                                        (pymeasure.instruments.hp.HP8753E
                                            (pymea-
                                                      scan_points
        sure.instruments.rohdeschwarz.hmp.HMP4040
                                                               property), 309
         method), 543
                                                      scan_single()
                                                                                                   (pymea-
save_setup()
                    (pymeasure.instruments.srs.SR830
                                                               sure.instruments.agilent.Agilent8722ES
        method), 578
                                                               method), 152
                                                                        (pymeasure.instruments.hp.HP8753E
save_state()
                                            (рутеа-
                                                      scan_single()
         sure.instruments.hp.hp856Xx.HP856Xx
                                                               method), 309
                                                                                                   (рутеа-
        method), 277
                                                      scan_start()
                                                               sure.instruments.keithley.KeithleyDMM6500
save_trace()
                                            (pymea-
        sure.instruments.hp.hp856Xx.HP856Xx
                                                               method), 391
                                                      scan_stop() (pymeasure.instruments.keithley.KeithleyDMM6500
        method), 287
save_var() (pymeasure.instruments.agilent.agilent4156.Agilent4156method), 391
        method), 168
                                                      scan_vch_end
                                                                                                   (pymea-
save_waveform()
                                             (pymea-
                                                               sure.instruments.keithley.KeithleyDMM6500
         sure.instruments.keysight.keysight81160A.Keysight81160AGhapperty), 391
```

scan_vch_start	(рутеа-	517		
sure.instruments.keithley.KeithleyD	MM6500	send_trigger()		(pymea-
property), 392		sure.inst	ruments.agilent.agilentB1500.	AgilentB1500
scanned_data()	(pymea-	method)		
sure.instruments.keithley.KeithleyD	<i>MM6500</i>	sense_mode(pym	easure.instruments.anritsu.An	ritsuMS2090A
method), 392		property		
ScannerCard2000Channel (class in	рутеа-	sense_mode(pym	easure.instruments.keithley.Ke	eithleyDAQ6510
sure.instruments.keithley.keithleyDl	MM6500),	property	), 403	
393		sensitivity(pyr	neasure.instruments.ametek.A	metek7270
ScientificInput (class in pymeasure.disp	olay.inputs),	property	), 221	
78		sensitivity(pyr	neasure.instruments.signalrec	overy.DSP7225
<pre>scpi_version (pymeasure.instruments.hp</pre>	.HP34401A	property		
property), 265		sensitivity(py	neasure.instruments.signalrec	overy.DSP7265
SCPIMixin (class in	рутеа-	property	), 562	
sure.instruments.generic_types), 10	8	sensitivity (p	measure.instruments.srs.SR51	10 prop-
<pre>screen_layout (pymeasure.instrument</pre>	s.srs.SR860	erty), 57	3	
property), 583		sensitivity (p	measure.instruments.srs.SR57	70 prop-
<pre>screenshot() (pymeasure.instrument</pre>	s.srs.SR860	erty), 57	4	
method), 583		sensitivity (p	measure.instruments.srs.SR83	80 prop-
SDS1072CML (class in	рутеа-	erty), 57	8	
sure.instruments.siglenttechnologies	s), 548	sensitivity (p	measure.instruments.srs.SR86	60 prop-
search_peak()	(pymea-	erty), 58	3	
sure.instruments.hp.hp856Xx.HP85	6Xx	sensitivity(pyr	neasure.instruments.yokogawa	a.aq6370series.AQ6370Ser
method), 289		property	), 641	
seed_voltage	(pymea-	sensitivity_le	vel	(рутеа-
sure.instruments.aculight.argos.Arg	os prop-	sure.inst	ruments.yokogawa.aq6370seri	es.AQ6370E
<i>erty</i> ), 116		property	), 643	
<pre>select_channel()</pre>	(pymea-	sensitvity (py	measure.instruments.srs.SR86	0 prop-
sure.instruments.rohdeschwarz.fsse	ries.FSW	erty), 58	3	
method), 542		sensor (pymeasur	e.instruments.lakeshore.lakesh	hore_base.LakeShoreTemp
<pre>select_for_output()</pre>	(pymea-	property	), 454	
sure.instruments.advantest.advantes	stR624X.SMU	U <b>Skansod_</b> 1 (pymea.	sure.instruments.inficon.sqm16	60.SQM160
method), 129		attribute	), 320	
selected_channel	(pymea-	sensor_2 (pymea.	sure.instruments.inficon.sqm16	60.SQM160
sure.instruments.rohdeschwarz.hmp	.HMP4040	attribute	), 320	
property), 543		sensor_3 (pymea	sure.instruments.inficon.sqm16	60.SQM160
selected_channel	(pymea-	attribute	), 320	
sure.instruments.siglenttechnologies	s.siglent_spd	ba <b>ssenSADB41.(p</b> ymea	sure.instruments.inficon.sqm16	60.SQM160
property), 545		attribute	), 320	
selected_channel_active	(pymea-	sensor_5 (pymea	sure.instruments.inficon.sqm16	60.SQM160
sure.instruments.rohdeschwarz.hmp	.HMP4040	attribute		~
property), 543			sure.instruments.inficon.sqm16	60.SQM160
self_calibrate()	(pymea-	attribute	v	~
sure.instruments.ni.virtualbench.Vii	* *		* *	(pymea-
method), 484			ruments.hp.HP437B method),	
<pre>self_test(pymeasure.instruments.advantes.</pre>	t.advantestR0		*	(рутеа-
property), 126			ruments.hp.HP437B method),	
self_test_result	(рутеа-	sensor_data_re	- · · · · · · · · · · · · · · · · · · ·	(рутеа-
sure.instruments.hp.HP34401A	property),		ruments.hp.HP437B method),	* *
265	r · r - · · // //			(pymea-
selftest() (pymeasure.instruments.ptw.p	otwUNIDOS		ruments.hp.HP437B method),	4.7
method), 517	, 51,1200	sensor_data_wr		(pymea-
selftest_result	(рутеа-		ruments.hp.HP437B method),	
sure.instruments.ptw.ptwUNIDOS	property),	sensor_serial	<sub>T</sub> 2	(pymea-
T	1 1 7/7			¥ -

$sure.instruments.thy racont.smartline\_v2.Smartli$	
property), 631	sure.instruments.lakeshore.LakeShore421
sensor_type (pymeasure.instruments.hp.HP437B prop-	property), 453
erty), 314	serial_number (pymea-
SensorChannel (class in pymea-	sure.instruments.mksinst.mks974b.MKS974B
sure.instruments.inficon.sqm160), 321	property), 476
sequence (pymeasure.instruments.rohdeschwarz.hmp.HM property), 543	IP\$⊕#@al_number (pymeasure.instruments.novanta.Fpu60 property), 492
	serial_number (pymea-
	antestR624Xsure.instruments.ptw.ptwDIAMENTOR prop-
method), 121	erty), 513
	serial_number (pymea-
sure.instruments.advantest.advantestR624X.Adv	
property), 121	517
	serial_parity (pymea-
	antestR624Xsure.instruments.rohdeschwarz.sfm.SFM
method), 121	property), 535
SequenceDialog (class in pymea-	serial_stopbits (pymea-
sure.display.widgets.sequencer_widget),	sure.instruments.rohdeschwarz.sfm.SFM
86	property), 535
SequenceInterruptionType (class in pymea-	SerialAdapter (class in pymeasure.adapters), 51
sure.instruments.advantest.advantestR624X),	series_resistance (pymea-
140	sure.instruments.agilent.agilent4156.SMU
SequencerTreeModel (class in pymea-	property), 169
sure.display.widgets.sequencer_widget),	series_resistor (pymea-
86	sure.instruments.agilent.agilentB1500.SMU
SequencerTreeView (class in pymea-	property), 190
sure.display.widgets.sequencer_widget),	service_request_enable_bits (pymea-
87	sure.instruments.anritsu.AnritsuMS464xB
SequencerWidget (class in pymea-	property), 241
sure.display.widgets.sequencer_widget),	
87	sure.instruments.advantest.advantestR624X.AdvantestR624X
serial (pymeasure.instruments.mksinst.mks937b.MKS93	
property), 474	SESR (class in pymea-
serial (pymeasure.instruments.tcpowerconversion.CXN	sure.instruments.advantest.advantestR624X),
property), 588	142
serial(pymeasure.instruments.tdk.tdk_gen40_38.TDK_0	
property), 592	sure.instruments.hp.hp856Xx.HP856Xx
serial(pymeasure.instruments.tdk.tdk_gen80_65.TDK_0	
property), 597	set_averaging() (pymea-
${\tt serial} \ (py measure. instruments. toptica. ibeams mart. IBeams $	nSmart sure.instruments.agilent.Agilent8722ES
property), 635	method), 152
serial_baud(pymeasure.instruments.rohdeschwarz.sfm	SFSMt_buffer() (pymea-
property), 534	sure.instruments.signalrecovery.DSP7225
serial_bits(pymeasure.instruments.rohdeschwarz.sfm	SFM method), 555
property), 534	set_buffer() (pymea-
serial_flowcontrol (pymea-	sure.instruments.signalrecovery.DSP7265
sure.instruments.rohdeschwarz.sfm.SFM	method), 562
property), 534	<pre>set_channel_A_mode()</pre>
serial_nr (pymeasure.instruments.attocube.anc300.Axis	<del>-</del> '
	sure.instruments.ametek.Ametek7270 method),
property), 248	sure.instruments.ametek.Ametek/2/0 metnoa), 221
property), 248	221

```
set_color() (pymeasure.display.widgets.plot_widget.PlotWidget
                                                                                                   sure.instruments.agilent.Agilent8722ES
              method), 85
                                                                                                   method), 152
set_color() (pymeasure.display.widgets.tab widget.TabWiketetlevel_position_to_max()
                                                                                                   sure.instruments.yokogawa.aq6370series.AQ6370Series
              method), 88
set_color() (pymeasure.display.widgets.table_widget.TableWidget method), 641
             method), 91
                                                                                     set_linear_scale()
                                                                                                                                                           (рутеа-
set_comparison_limits()
                                                                      (pymea-
                                                                                                   sure.instruments.hp.hp856Xx.HP856Xx
             sure.instruments.advantest.advantestR624X.SMUChannel method), 275
             method), 138
                                                                                     set_lo_common_connection_relay()
                                                                                                                                                           (pymea-
                                                                                                   sure.instruments.advantest.advantestR624X.AdvantestR624X
set_continuous_sensor_transition()
                                                                      (рутеа-
             sure.instruments.thyracont.smartline_v2.SmartlineV2
                                                                                                   method), 126
             method), 631
                                                                                     set_low() (pymeasure.instruments.thyracont.smartline_v2.SmartlineV2
set_continuous_sweep
                                                                      (pymea-
                                                                                                   method), 631
             sure.instruments.hp.hp856Xx.HP856Xx
                                                                                     set_marker_delta_to_span()
                                                                              at-
                                                                                                                                                           (pymea-
             tribute), 293
                                                                                                   sure.instruments.hp.hp856Xx.HP856Xx
set_crt_adjustment_pattern()
                                                                      (рутеа-
                                                                                                   method), 291
             sure.instruments.hp.hp856Xx.HP856Xx
                                                                                     set_marker_minimum()
                                                                                                                                                           (pymea-
             method), 280
                                                                                                   sure.instruments.hp.hp856Xx.HP856Xx
set_current_mode()
                                                                      (pymea-
                                                                                                   method), 290
             sure.instruments.ametek.Ametek7270 method),
                                                                                     set_marker_to_center_frequency()
                                                                                                                                                           (pymea-
              221
                                                                                                   sure.instruments.hp.hp856Xx.HP856Xx
set_default_sensor_transition()
                                                                      (рутеа-
                                                                                                   method), 289
             sure.instruments.thyracont.smartline_v2.Smartlineset_marker_to_center_frequency_step_size()
                                                                                                    (pymeasure.instruments.hp.hp856Xx.HP856Xx
             method), 631
set_defaults()
                                                                      (pymea-
                                                                                                   method), 291
             sure. instruments. parker. Parker GV6
                                                                     method),
                                                                                     set_marker_to_reference_level()
                                                                                                                                                           (рутеа-
                                                                                                   sure.instruments.hp.hp856Xx.HP856Xx
set_differential_mode()
                                                                      (рутеа-
                                                                                                   method), 291
             sure.instruments.ametek.Ametek7270 method),
                                                                                     set_max_over_voltage()
                                                                                                                                                           (pymea-
                                                                                                   sure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38
              221
                                                                                                   method), 592
set_digital_output()
                                                                      (pymea-
             sure.instruments.advantest.advantestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.AdvansestR624X.Advansest
                                                                                                                                                            (рутеа-
                                                                                                   sure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65
             method), 120
set_direct_sensor_transition()
                                                                      (pymea-
                                                                                                   method), 597
             sure.instruments.thyracont.smartline_v2.Smartlineset_maximum_hold
                                                                                                                                                           (pymea-
                                                                                                   sure.instruments.hp.hp856Xx.HP856Xx
             method), 631
                                                                                                                                                                   at-
set_field() (pymeasure.instruments.oxfordinstruments.IPS120 10 tribute), 278
             method), 500
                                                                                     set_minimum_hold
                                                                                                                                                           (pymea-
set_fixed_frequency()
                                                                      (рутеа-
                                                                                                    sure.instruments.hp.hp856Xx.HP856Xx
             sure.instruments.agilent.Agilent8722ES
                                                                                                   tribute), 278
             method), 153
                                                                                     set_model() (pymeasure.display.widgets.table widget.Table
set_fixed_frequency()
                                                                      (pymea-
                                                                                                   method), 90
              sure.instruments.hp.HP8753E method), 309
                                                                                     set_monitored_quantity()
                                                                                                                                                           (pymea-
                                                                                                   sure.instruments.hcp.TC038 method), 261
set_full_span()
                                                                      (pymea-
             sure.instruments.hp.hp856Xx.HP856Xx
                                                                                     set_output_format()
                                                                                                                                                           (рутеа-
                                                                                                   sure.instruments.advantest.advantestR624X.AdvantestR624X
              method), 283
set_fullband() (pymeasure.instruments.hp.HP8561B
                                                                                                   method), 122
             method), 299
                                                                                     set_output_type()
                                                                                                                                                           (pymea-
                                                                                                   sure.instruments.advantest.advantestR624X.SMUChannel
set_hardware_limits()
                                                                      (рутеа-
              sure.instruments.parker.ParkerGV6
                                                                     method),
                                                                                                   method), 127
                                                                                     set_parameter()
                                                                                                                                                           (рутеа-
set_high() (pymeasure.instruments.thyracont.smartline v2.SmartlineW2.display.inputs.BooleanInput
                                                                                                                                                          method),
              method), 631
set_IF_bandwidth()
                                                                      (pymea- set_parameter()
                                                                                                                         (pymeasure.display.inputs.Input
```

method), 77		<pre>set_timed_arm()</pre>	(pymea-
set_parameter()	(рутеа-	sure.instruments.keithley.Keithley2	
sure.display.inputs.IntegerInput	method),	method), 349	2400
78	memou),	set_timed_arm()	(рутеа-
<pre>set_parameter() (pymeasure.display.inputs</pre>	s I istInnut	sure.instruments.keithley.Keithley	* *
method), 78	изипри	method), 375	7221
set_parameter()	(nymaa	set_timing_parameters()	(pymea-
sure.display.inputs.ScientificInput	method),	sure.instruments.advantest.advant	* *
78	memoa),		estR024A.SMOChannet
, -	(m)maa	method), 129 set_title() (pymeasure.instruments.hp.hp	056V, UD056V,
set_parameters()		**	0000Ax.HF 000Ax
sure.display.windows.managed_windo	ow.manage		(22,444, 0.0)
method), 94	(	set_trace_data_a	(pymea-
set_parameters()	(pymea-	sure.instruments.hp.hp856Xx.HP8	56Xx at-
sure.experiment.procedure.Procedure	e method),	tribute), 286	
66	EL 1 E2 (1	set_trace_data_b	(pymea-
set_point (pymeasure.instruments.fluke.l	Fluke7341	sure.instruments.hp.hp856Xx.HP8	56Xx at-
property), 256		tribute), 286	
set_point_number	4.0	set_trigger_config()	(рутеа-
sure.instruments.temptronic.ATSBase	prop-	sure.instruments.siglenttechnologi	es.siglent_sds1072cml.Trigger <b>(</b>
erty), 617		method), 550	
<pre>set_ramp_delay()</pre>	(pymea-	set_trigger_counts()	(pymea-
sure.instruments.danfysik.Danfysik85	00	sure.instruments.keithley.Keithley2	2400
method), 250		method), 349	
<pre>set_ramp_to_current()</pre>	(pymea-	set_voltage_mode()	(рутеа-
sure.instruments.danfysik.Danfysik85	00	sure.instruments.ametek.Ametek72	(70 method),
method), 250		221	
set_reference_mode()	(pymea-	set_voltage_mode()	(рутеа-
sure.instruments.ametek.Ametek7270		sure.instruments.signalrecovery.D	SP7225
221	,,,	method), 556	
<pre>set_sample_mode()</pre>	(рутеа-	set_voltage_mode()	(рутеа-
		JChannel sure.instruments.signalrecovery.D	* *
method), 129		method), 562	
set_scaling()	(nymea-	set_wire_mode()	(рутеа-
sure.instruments.spellmanhv.Spellman		sure.instruments.advantest.advant	4.5
method), 566	12111	method), 137	esino2+M.SM o Charinei
set_scaling() (pymeasure.instruments.	crc SR830		(рутеа-
method), 578	373.511050	sure.instruments.signalrecovery.D	A .
set_scanner_control()	(nymaa	method), 555	31 /223
sure.instruments.advantest.advantest	(pymea- 2624V SMI		(pymag
	1024A.SMC		(pymea-
method), 128	(*************	sure.instruments.signalrecovery.D	SF / 203
set_sequence()	(pymea-	method), 562	
sure.instruments.danfysik.Danfysik85	00	setData() (pymeasure.display.widgets.sequ	iencer_wiaget.Sequencer1reeM
method), 251	C	method), 87	
set_signal_identification_to_center_			(pymea-
(pymeasure.instruments.hp.HP8561B	method),	sure.instruments.signalrecovery.D	SP/225
299	,	method), 555	
<pre>set_software_limits()</pre>	(pymea-	setDifferentialMode()	(pymea-
sure.instruments.parker.ParkerGV6	method),	sure.instruments.signalrecovery.D	SP/265
506		method), 562	
set_sweep_time_fastest()	(рутеа-	setEditorData()	(рутеа-
sure.instruments.hp.HP8753E method		sure.display.widgets.sequencer_wi	dget.ComboBoxDelegate
set_temperature()	(pymea-	method), 85	
sure.instruments.temptronic.ATSB ase	method),	setEditorData()	(рутеа-
618		sure.display.widgets.sequencer_wi	dget.LineEditDelegate

```
method), 85
                                                       shape (pymeasure.instruments.activetechnologies.AWG401x.ChannelAFG
setModel() (pymeasure.display.widgets.sequencer_widget.Sequencerplinepolity), 115
         method), 87
                                                       shape
                                                                (pymeasure.instruments.agilent.Agilent33220A
setModel() (pymeasure.display.widgets.table_widget.Table
                                                                property), 174
         method), 90
                                                       shape
                                                                  (pymeasure.instruments.agilent.Agilent33500
setModelData()
                                             (pymea-
                                                                property), 177
         sure.display.widgets.sequencer widget.ComboBoxshdoe(wymeasure.instruments.agilent.agilent33500.Agilent33500Channel
         method), 85
                                                                property), 180
setModelData()
                                             (pymea-
                                                       shape (pymeasure.instruments.hp.HP33120A property),
         sure.display.widgets.sequencer_widget.LineEditDelegate
         method), 86
                                                       shape (pymeasure.instruments.hp.HP8116A property),
setpoint (pymeasure.instruments.aja.DCXS property),
                                                                273
                                                       shape (pymeasure.instruments.keysight.Keysight81160A
setpoint (pymeasure.instruments.hcp.TC038 property),
                                                                property), 438
                                                       shape(pymeasure.instruments.rigol.rigol_dg800.VoltageChannel
setpoint (pymeasure.instruments.hcp.TC038D prop-
                                                                property), 525
                                                       shape (pymeasure.instruments.tektronix.afg3152c.AFG3152CChannel
         erty), 261
setpoint (pymeasure.instruments.lakeshore_lakeshore_base.LakeShoprebpeaten)Channel
                                                       shield(pymeasure.instruments.signalrecovery.DSP7225
         property), 455
setpoint (pymeasure.instruments.mksinst.mksinst.RelayChannel
                                                                property), 556
                                                       \verb|shield| (pymeasure. instruments. signal recovery. DSP7265
         property), 472
setpoint(pymeasure.instruments.proterial.rod4.ROD4Channel
                                                                property), 562
                                                       shield_to_guard_enabled
         property), 511
                                                                                                    (pymea-
setpoint (pymeasure.instruments.tcpowerconversion.CXN
                                                                sure.instruments.keithley.Keithley6221
                                                                                                      prop-
         property), 588
                                                                erty), 375
setpoint (pymeasure.instruments.thermotron.Thermotron3sh00rt_enabled
                                                                                                    (pymea-
                                                                sure.instruments.agilent.agilent4284A.Agilent4284ACorrection
         property), 626
setting() (pymeasure.instruments.common_base.CommonBase
                                                                property), 200
         static method), 102
                                                       shutdown() (pymeasure.experiment.procedure.Procedure
setting() (pymeasure.instruments.keysight.Keysight81160A
                                                                method), 66
         static method), 438
                                                       shutdown()
                                                                       (pymeasure.experiment.workers.Worker
setting() (pymeasure.instruments.keysight.KeysightE36312A
                                                                method), 72
         static method), 419
                                                       shutdown() (pymeasure.instruments.agilent.Agilent8257D
setting() (pymeasure.instruments.keysight.KeysightE3631A
                                                                method), 151
         static method), 427
                                                       shutdown() (pymeasure.instruments.ametek.Ametek7270
setting() (pymeasure.instruments.lecroy.LeCroyT3DSO1204
                                                                method), 221
         static method), 465
                                                       shutdown()
                                                                         (pymeasure.instruments.ami.AMI430
setup() (pymeasure.instruments.teledyne.teledyne_oscilloscope.TeledynethOddjllQscopeChannel
         method), 609
                                                       shutdown() (pymeasure.instruments.andeenhagerling.AH2700A
setup() (pymeasure.instruments.teledyne.teledyneMAUI.TeledyneMAbblathdnyn229
                                                       shutdown() (pymeasure.instruments.anritsu.AnritsuMG3692C
         method), 612
setup_parser()
                                             (pymea-
                                                                method), 230
                                                       shutdown() (pymeasure.instruments.deltaelektronika.SM7045D
         sure.display.console.ConsoleArgumentParser
         method), 75
                                                                method), 253
                     (pymeasure.display.plotter.Plotter
setup_plot()
                                                       shutdown() (pymeasure.instruments.eurotest.EurotestHPP120256
         method), 81
                                                                method), 255
setup_temperature()
                                             (pymea-
                                                       shutdown() (pymeasure.instruments.fwbell.FWBell5080
         sure.instruments.keithley.keithley2182.Keithley2182Channelmethod), 259
         method), 399
                                                       shutdown()
                                                                         (pymeasure.instruments.hp.HP8116A
setup_voltage()
                                             (рутеа-
                                                                method), 273
         sure.instruments.keithley.keithley2182.Keithley2182hdh.dowd()
                                                                         (pymeasure.instruments.hp.HP8657B
                                                                method), 307
         method), 399
SFM (class in pymeasure.instruments.rohdeschwarz.sfm), shutdown()
                                                                         (pymeasure.instruments.hp.HP8753E
                                                                method), 309
         526
```

```
shutdown() (pymeasure.instruments.hp.HPLegacyInstrumeshutdown() (pymeasure.instruments.ni.virtualbench.VirtualBench.MixedS
         method), 316
                                                                method), 488
shutdown()
                    (pymeasure.instruments.Instrument shutdown() (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerS
                                                                method), 489
         method), 104
shutdown() (pymeasure.instruments.keithley.Keithley2000 shutdown()
                                                                      (pymeasure.instruments.proterial.ROD4
        method), 328
                                                                method), 510
shutdown() (pymeasure.instruments.keithley.Keithley2182 shutdown() (pymeasure.instruments.siglenttechnologies.siglent spdbase.S
         method), 397
                                                                method), 546
shutdown() (pymeasure.instruments.keithley.Keithley2200 shutdown() (pymeasure.instruments.signalrecovery.DSP7225
         method), 334
                                                                method), 556
shutdown() (pymeasure.instruments.keithley.Keithley2260Bhutdown() (pymeasure.instruments.signalrecovery.DSP7265
         method), 336
                                                                method), 563
shutdown() (pymeasure.instruments.keithley.Keithley.281\$hutdown() (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38
                                                                method), 592
        method), 339
shutdown() (pymeasure.instruments.keithley.Keithley2306 shutdown() (pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65
         method), 343
                                                                method), 597
shutdown() (pymeasure.instruments.keithley.Keithley2400 shutdown() (pymeasure.instruments.teledyne.TeledyneT3AFG
         method), 349
                                                                method), 601
shutdown() (pymeasure.instruments.keithley.Keithley2450 shutdown() (pymeasure.instruments.temptronic.ATSBase
        method), 356
                                                                method), 618
shutdown() (pymeasure.instruments.keithley.Keithley2510 shutdown() (pymeasure.instruments.texio.TexioPSW360L30
        method), 361
                                                                method), 624
shutdown() (pymeasure.instruments.keithley.Keithley2600 shutdown() (pymeasure.instruments.toptica.ibeamsmart.IBeamSmart
         method), 364
                                                                method), 635
shutdown() (pymeasure.instruments.keithley.Keithley2700 shutdown() (pymeasure.instruments.yokogawa.Yokogawa7651
        method), 367
                                                                method), 638
shutdown() (pymeasure.instruments.keithley.Keithley2750 shutter_delay
                                                                            (pymeasure.instruments.aja.DCXS
         method), 370
                                                                property), 219
shutdown() (pymeasure.instruments.keithley.Keithley6221 shutter_open (pymeasure.instruments.novanta.Fpu60
         method), 375
                                                                property), 492
shutdown() (pymeasure.instruments.keithley.Keithley6517Bhutter_state
                                                                            (pymeasure.instruments.aja.DCXS
         method), 381
                                                                property), 219
shutdown() (pymeasure.instruments.keithley.KeithleyDAQGIkhal_identification
                                                                                                    (рутеа-
         method), 403
                                                                sure.instruments.hp.HP8561B
                                                                                                  property),
shutdown() (pymeasure.instruments.keysight.Keysight81160A
                                                       signal_identification_frequency
        method), 438
                                                                                                    (pymea-
shutdown() (pymeasure.instruments.keysight.KeysightE36312A
                                                                sure.instruments.hp.HP8561B
                                                                                                  property),
         method), 419
shutdown()(pymeasure.instruments.keysight.KeysightE363shAgnal_inverted
                                                                           (pymeasure.instruments.srs.SR570
                                                                property), 574
         method), 427
shutdown() (pymeasure.instruments.lakeshore.LakeShore4\( \frac{1}{2} \) ignalChannel
                                                                                                    рутеа-
                                                                              (class
                                                                                           in
         method), 453
                                                                sure.instruments.teledyne.teledyneT3AFG),
shutdown() (pymeasure.instruments.lecroy.LeCroyT3DSO1204
                                                       sine (pymeasure.instruments.rigol.rigol\_dg800.VoltageChannel
        method), 465
shutdown()
              (pymeasure.instruments.newport.ESP300
                                                                property), 525
                                                                                                    (рутеа-
                                                       sine_amplitudepreset1
         method), 477
shutdown() (pymeasure.instruments.ni.virtualbench. VirtualBench sure.instruments.srs.SR860 property), 583
                                                       sine_amplitudepreset2
         method), 491
                                                                                                    (pymea-
shutdown() (pymeasure.instruments.ni.virtualbench.VirtualBench.DisginelInstattanteptstsrs.SR860 property), 583
                                                       sine_amplitudepreset3
         method), 479
                                                                                                    (рутеа-
shutdown() (pymeasure.instruments.ni.virtualbench.VirtualBench.DigitælMsdhimætets.srs.SR860 property), 583
        method), 481
                                                      sine_amplitudepreset4
                                                                                                    (pymea-
shutdown() (pymeasure.instruments.ni.virtualbench.VirtualBench.Fusuoticintsteuments.srs.SR860 property), 583
         method), 484
                                                       sine_dclevelpreset1
                                                                                                    (pymea-
```

	CMII (-I : :
sure.instruments.srs.SR860 property), 583	SMU (class in pymea-
sine_dclevelpreset2 (pymea- sure.instruments.srs.SR860 property), 583	sure.instruments.agilent.agilent4156), 169 SMU (class in pymea-
sine_dclevelpreset3 (pymea-	SMU (class in pymea- sure.instruments.agilent.agilentB1500), 189
sure.instruments.srs.SR860 property), 583	SMU (class in pymea-
sine_dclevelpreset4 (pymea-	sure.instruments.keithley.keithley4200), 405
sure.instruments.srs.SR860 property), 583	SMU_MEASUREMENT (pymea-
sine_voltage (pymeasure.instruments.srs.SR830 prop-	sure.instruments.agilent.agilentB1500.WaitTimeType
erty), 578	attribute), 197
	smu_names (pymeasure.instruments.agilent.agilentB1500.AgilentB1500
erty), 583	property), 185
single() (pymeasure.instruments.keysight.KeysightDSOX	
method), 409	sure.instruments.agilent.agilentB1500.AgilentB1500
single() (pymeasure.instruments.lecroy.LeCroyT3DSO12	
method), 466	SMU_SOURCE (pymeasure.instruments.agilent.agilentB1500.WaitTimeType
single() (pymeasure.instruments.teledyne.TeledyneOscill	
method), 604	SMUChannel (class in pymea-
single_sweep() (pymea-	sure.instruments.advantest.advantestR624X),
sure.instruments.anritsu.AnritsuMS9710C	127
method), 232	SMUChannel (class in pymea-
single_sweep() (pymea-	sure.instruments.agilent.agilentE5270B),
sure.instruments.rohdeschwarz.fsseries.FSSeries	s 214
method), 540	SMUCurrentRanging (class in pymea-
<pre>sizeHint() (pymeasure.display.widgets.image_widget.Im</pre>	nageWidget sure.instruments.agilent.agilentB1500), 193
method), 84	SMUVoltageRanging (class in pymea-
<pre>sizeHint() (pymeasure.display.widgets.plot_widget.Plot</pre>	
method), 85	sn (pymeasure.instruments.hp.HP8753E property), 309
	O <b>£2024) Ly CpyyTe318180.12814 Ghant</b> ækrs.SR830 method), 578
property), 470	snap() (pymeasure.instruments.srs.SR860 method), 583
property), 470 slew_rate(pymeasure.instruments.danfysik.Danfysik8500	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method),
property), 470 slew_rate(pymeasure.instruments.danfysik.Danfysik8500 property), 251	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584
property), 470 slew_rate(pymeasure.instruments.danfysik.Danfysik850cproperty), 251 slew_rate_1(pymeasure.instruments.razorbill.razorbillR	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$\text{SOD}\text{tware_version} (pymeasure.instruments.aja.DCXS
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$00tware_version (pymea-
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik850c property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RPSOOTware_version (pymeasure.instruments.aja.DCXS property), 220 RPSOOTware_version (pymeasure.instruments.aja.pymeasure.instruments.novanta.Fpu60 property),
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik850c property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 prop-	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RPSODtware_version (pymeasure.instruments.aja.DCXS property), 220 RPSODtware_version (pymeasure.instruments.aja.DCXS property), 240 sure.instruments.novanta.Fpu60 property), 493
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik850c property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$00tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.novanta.Fpu60 property), 493
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik850c property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 prop-	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RPSODtware_version (pymeasure.instruments.aja.DCXS property), 220 RPSODtware_version (pymeasure.instruments.aja.DCXS property), 240 sure.instruments.novanta.Fpu60 property), 493
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$00tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556 slope (pymeasure.instruments.signalrecovery.DSP7265	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$00tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RPSOOTWARE_version (pymeasure.instruments.aja.DCXS property), 220 RPSOOTWARE_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik850c property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556 slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$00tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.racal.ra
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik850c property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556 slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563 slot (pymeasure.instruments.thorlabs.ThorlabsPro8000)	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00\text{tware_version} (pymeasure.instruments.aja.DCXS property), 220 RP\$00\text{tware_version} (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556 slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563 slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$00tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556 slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563 slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627 SM7045D (class in pymea-	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00\text{Ptware_version} (pymeasure.instruments.aja.DCXS property), 220 RP\$00\text{tware_version} (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535
property), 470 slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251 slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521 slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522 slope (pymeasure.instruments.ametek.Ametek7270 property), 221 slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556 slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563 slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627 SM7045D (class in pymeasure.instruments.deltaelektronika), 252	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$00\text{Ptware_version} (pymeasure.instruments.aja.DCXS property), 220 RP\$00\text{tware_version} (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.keithley.Keithley6221 prop-
property), 470  slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251  slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521  slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522  slope (pymeasure.instruments.ametek.Ametek7270 property), 221  slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556  slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563  slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627  SM7045D (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.thyracont.smartline_v1), 627	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$000tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$1000tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.keithley.Keithley6221 property), 375
property), 470  slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251  slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521  slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522  slope (pymeasure.instruments.ametek.Ametek7270 property), 221  slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556  slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563  slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627  SM7045D (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.thyracont.smartline_v1),	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$000tware_version (pymeasure.instruments.aja.DCXS property), 220 RP\$000tware_version (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.keithley.Keithley6221 property), 375 source_compliance (pymeasure.compliance)
property), 470  slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251  slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521  slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522  slope (pymeasure.instruments.ametek.Ametek7270 property), 221  slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556  slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563  slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627  SM7045D (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v2),	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$\text{S00}\text{tware_version}  (pymeasure.instruments.aja.DCXS  property), 220 RP\$\text{S00}\text{tware_version}  (pymeasure.instruments.novanta.Fpu60  property), 493 software_version  (pymeasure.instruments.racal.Racal1992  property), 520 Sound_Channel  (class  in  pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode \((pymeasure.instruments.rohdeschwarz.sfm.SFM  property), 535 source_auto_range  (pymeasure.instruments.keithley.Keithley6221  property), 375 source_compliance  (pymeasure.instruments.keithley.Keithley6221  property), 375 source_current  (pymea-
property), 470  slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251  slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521  slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522  slope (pymeasure.instruments.ametek.Ametek7270 property), 221  slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556  slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563  slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627  SM7045D (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v2), 629	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$\text{OD}\text{tware_version} (pymeasure.instruments.aja.DCXS property), 220 RP\$\text{SOD}\text{tware_version} (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.keithley.Keithley6221 property), 375 source_compliance (pymeasure.instruments.keithley.Keithley6221 property), 375 source_current (pymeasure.instruments.keithley.Keithley2400 property).
property), 470  slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251  slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521  slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522  slope (pymeasure.instruments.ametek.Ametek7270 property), 221  slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556  slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563  slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627  SM7045D (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v2), 629  SmartlineV2.Sources (class in pymeasure.instruments.thyracont.smartline_v2), 629	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$\textit{OD}\text{tware_version} (pymeasure.instruments.aja.DCXS property), 220 RP\$\textit{OD}\text{tware_version} (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.keithley.Keithley6221 property), 375 source_compliance (pymeasure.instruments.keithley.Keithley6221 property), 375 source_compliance (pymeasure.instruments.keithley.Keithley6221 property), 375 source_current (pymeasure.instruments.keithley.Keithley2400 property), 349
property), 470  slew_rate (pymeasure.instruments.danfysik.Danfysik8500 property), 251  slew_rate_1 (pymeasure.instruments.razorbill.razorbillR property), 521  slew_rate_2 (pymeasure.instruments.razorbill.razorbillR property), 522  slope (pymeasure.instruments.ametek.Ametek7270 property), 221  slope (pymeasure.instruments.signalrecovery.DSP7225 property), 556  slope (pymeasure.instruments.signalrecovery.DSP7265 property), 563  slot (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 627  SM7045D (class in pymeasure.instruments.deltaelektronika), 252  SmartlineV1 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v1), 627  SmartlineV2 (class in pymeasure.instruments.thyracont.smartline_v2), 629	snap() (pymeasure.instruments.srs.SR860 method), 583 00 snap_all() (pymeasure.instruments.srs.SR860 method), 584 RP\$\text{OD}\text{tware_version} (pymeasure.instruments.aja.DCXS property), 220 RP\$\text{SOD}\text{tware_version} (pymeasure.instruments.novanta.Fpu60 property), 493 software_version (pymeasure.instruments.racal.Racal1992 property), 520 Sound_Channel (class in pymeasure.instruments.rohdeschwarz.sfm), 536 sound_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 535 source_auto_range (pymeasure.instruments.keithley.Keithley6221 property), 375 source_compliance (pymeasure.instruments.keithley.Keithley6221 property), 375 source_current (pymeasure.instruments.keithley.Keithley2400 property).

erty), 357	property), 381
source_current (pymea-	source_enabled (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.yokogawa.Yokogawa7651
erty), 375	property), 638
source_current (pymea-	source_enabled (pymea-
sure.instruments.yokogawa.Yokogawa7651	sure.instruments.yokogawa.YokogawaGS200
property), 638	property), 639
	source_level (pymea-
sure.instruments.keithley.Keithley2450 property), 357	sure.instruments.yokogawa.YokogawaGS200 property), 639
source_current_delay_auto (pymea-	source_leveling_control (pymea-
sure.instruments.keithley.Keithley2450 property), 357	sure.instruments.hp.HP8560A property), 296
source_current_range (pymea-	source_low_state (pymea-
sure.instruments.keithley.Keithley2400 property), 349	sure.instruments.agilent.AgilentB2985 property), 210
source_current_range (pymea-	source_mode(pymeasure.instruments.keithley.Keithley2400
sure.instruments.keithley.Keithley2450 prop-	property), 349
erty), 357	source_mode (pymeasure.instruments.keithley.Keithley2450
source_current_range (pymea-	property), 357
sure.instruments.yokogawa.Yokogawa7651 property), 638	source_mode (pymeasure.instruments.yokogawa.Yokogawa7651 property), 638
source_current_resistance_limit (pymea-	$\verb source_mode  (pymeasure.instruments.yokogawa.YokogawaGS200 $
sure.instruments.keithley.Keithley6517B	property), 639
property), 381	source_off_state (pymea-
source_delay (pymea-	sure.instruments.agilent.AgilentB2985 prop-
sure.instruments.keithley.Keithley2400 prop-	erty), 210
erty), 349	source_power (pymeasure.instruments.hp.HP8560A
source_delay (pymea-	property), 296
sure.instruments.keithley.Keithley6221 prop-	source_power_enabled (pymea-
erty), 375	sure.instruments.hp.HP8560A property),
source_delay_auto (pymea-	297
sure.instruments.keithley.Keithley2400 property), 349	source_power_offset (pymea- sure.instruments.hp.HP8560A property),
	sure.instruments.hp.HP8560A property), 297
source_enabled (pymea- sure.instruments.agilent.AgilentB2985 prop-	source_power_step (pymea-
erty), 210	sure.instruments.hp.HP8560A property),
source_enabled (pymea-	297
sure.instruments.bkprecision.BKPrecision9130B	
property), 248	sure.instruments.hp.HP8560A property),
source_enabled (pymea-	297
sure.instruments.keithley.Keithley2400 prop-	source_power_sweep_enabled (pymea-
erty), 349	sure.instruments.hp.HP8560A property),
source_enabled (pymea-	297
sure.instruments.keithley.Keithley2450 prop-	source_range (pymea-
erty), 357	sure.instruments.keithley.Keithley6221 prop-
source_enabled (pymea-	erty), 375
sure.instruments.keithley.Keithley2510 prop-	
erty), 361	source_range (pymea-
• •	sure.instruments.yokogawa.YokogawaGS200
source_enabled (pymea-	sure.instruments.yokogawa.YokogawaGS200 property), 639
source_enabled (pymea- sure.instruments.keithley.Keithley6221 prop-	sure.instruments.yokogawa.YokogawaGS200 property), 639 source_voltage (pymea-
source_enabled (pymea- sure.instruments.keithley.Keithley6221 prop- erty), 375	sure.instruments.yokogawa.YokogawaGS200 property), 639 source_voltage (pymea- sure.instruments.agilent.AgilentB2985 prop-
source_enabled (pymea- sure.instruments.keithley.Keithley6221 prop-	sure.instruments.yokogawa.YokogawaGS200 property), 639 source_voltage (pymea-

sure.instruments.keithley.Keithley2400 property), 349	sure.instruments.siglenttechnologies.siglent_spdbase), 546
source_voltage (pymea-	special_channel (pymea-
sure.instruments.keithley.Keithley2450 prop-	sure.instruments.rohdeschwarz.sfm.SFM
erty), 357	property), 535
source_voltage (pymea-	
sure.instruments.keithley.Keithley6517B	sure.instruments.racal.Racal1992 property),
property), 381	520
source_voltage (pymea-	
sure.instruments.yokogawa.Yokogawa7651	sure.instruments.racal.Racal1992 property),
property), 638	520
source_voltage_delay (pymea-	
sure.instruments.keithley.Keithley2450 prop-	sure.instruments.hp.hp856Xx.SweepCoupleMode
erty), 357	attribute), 303
sure.instruments.keithley.Keithley2450 prop-	sure.instruments.spellmanhv), 564
erty), 357	split_view(pymeasure.instruments.rohdeschwarz.fsseries.FSW
source_voltage_range (pymea-	property), 542
sure.instruments.agilent.AgilentB2985 prop-	splitPath() (pymeasure.display.widgets.filename_widget.PlaceholderCon
erty), 210	method), 83
source_voltage_range (pymea-	spol1() (pymeasure.instruments.philips.PM6669
sure.instruments.keithley.Keithley2400 prop-	method), 507
erty), 349	SPOT (pymeasure.instruments.agilent.agilentB1500.MeasMode
source_voltage_range (pymea-	attribute), 195
sure.instruments.keithley.Keithley2450 prop-	SQM160 (class in pymeasure.instruments.inficon.sqm160),
erty), 357	319
source_voltage_range (pymea-	
sure.instruments.keithley.Keithley6517B	property), 525
property), 381	square_dutycycle (pymea-
source_voltage_range (pymea-	sure.instruments.agilent.Agilent33220A
sure.instruments.yokogawa.Yokogawa7651	property), 174
property), 638	square_dutycycle (pymea-
SourceLevelingControlMode (class in pymea-	sure.instruments.agilent.Agilent33500 prop-
sure.instruments.hp.hp856Xx), 302	erty), 177
spacing (pymeasure.instruments.agilent.agilent4156.VAR	
property), 170	sure. instruments. a gilent. a gilent 33500. A gilent 33500 Channel
span (pymeasure.instruments.hp.hp856Xx.HP856Xx at-	property), 180
tribute), 283	square_dutycycle (pymea-
span_frequency (pymea-	sure.instruments.keysight.Keysight81160A
sure.instruments.advantest.advantestR3767CG.A	dvantestR3 <b>767<sub>1</sub>CC</b> ty), 438
property), 117	squelch (pymeasure.instruments.hp.hp856Xx.HP856Xx
<pre>span_frequency (pymeasure.instruments.hp.HP8753E</pre>	attribute), 281
property), 309	SR510 (class in pymeasure.instruments.srs), 573
SPD1168X (class in pymea-	SR570 (class in pymeasure.instruments.srs), 573
sure.instruments.siglenttechnologies), 548	SR830 (class in pymeasure.instruments.srs), 575
SPD1305X (class in pymea-	SR860 (class in pymeasure.instruments.srs), 579
sure.instruments.siglenttechnologies), 548	SRER (class in pymea-
SPDBase (class in pymea-	sure.instruments.advantest.advantestR624X),
sure.instruments.siglenttechnologies.siglent_spdl	
545	srq_enabled(pymeasure.instruments.advantest.advantestR624X.Advantes
SPDChannel (class in pymea-	property), 119
sure.instruments.siglenttechnologies.siglent_spdb	
546	property), 317
SPDSingleChannelBase (class in pymea-	srq_event_enabled (pymea-

sure.instruments.keithley.Keithley6221 prop	
erty), 375	sure.instruments.keithley.Keithley2450
SRQ_mask (pymeasure.instruments.hp.HP3437A prop	
erty), 266	start_buffer() (pymea-
SRQ_mask (pymeasure.instruments.hp.HP3478A prop	
erty), 268  SPONack (mymagsura instruments philips PM6660, processes)	method), 367
SRQMask (pymeasure.instruments.philips.PM6669 property), 506	
STAIRCASE_SWEEP (pymed	sure.instruments.keithley.Keithley6221 method), 376
sure.instruments.agilent.agilentB1500.MeasM	
attribute), 195	sure.instruments.keithley.Keithley6517B
staircase_sweep_source() (pymed	
sure.instruments.agilent.agilentB1500.SMU	
method), 191	sure.instruments.keithley.KeithleyDAQ6510
StaircaseSweepPostOutput (class in pymeo	
sure.instruments.agilent.agilentB1500), 196	
	sure.instruments.signalrecovery.DSP7225
sure.instruments.keithley.Keithley2182 prop	
erty), 397	start_buffer() (pymea-
standard_devs (pymed	
sure.instruments.keithley.Keithley2400 prop	·
erty), 349	start_frequency (pymea-
standard_devs (pymed	
sure.instruments.keithley.Keithley2450 prop	
erty), 357	start_frequency (pymea-
standard_event_enabled (pymed	the contract of the contract o
sure.instruments.keithley.Keithley6221 prop	
erty), 375	start_frequency (pymea-
standard_events (pymed	_ · · · · · · · · · · · · · · · · · · ·
sure.instruments.keithley.Keithley6221 prop	
erty), 376	start_frequency (pymea-
standby() (pymeasure.instruments.advantest.advantest	
method), 119	property), 153
<pre>standby() (pymeasure.instruments.advantest.advantest</pre>	R62\$Xa\$MLfChquerlcy (pymea-
method), 137	sure.instruments.agilent.AgilentE4408B
start (pymeasure.instruments.agilent.agilent4156.VAR	X property), 153
property), 171	start_frequency (pymea-
${\tt START}(py measure. instruments. a gilent. a gilent B1500. Starting and the starting a$	ircaseSweepP <b>sst@inptru</b> ments.agilent.agilentE5062A.VNAChannel
attribute), 197	property), 159
<pre>start() (pymeasure.experiment.experiment.Experiment</pre>	
method), 64	sure.instruments.hp.hp856Xx.HP856Xx at-
start() (pymeasure.instruments.temptronic.ATSBas	
method), 618	start_frequency (pymea-
start_autovernier() (pymed	
sure.instruments.hp.HP8116A method), 273	309
start_buffer() (pymed	- 1
sure.instruments.keithley.Keithley2000	property), 151
method), 328	start_ramp() (pymea-
start_buffer() (pymed	
sure.instruments.keithley.Keithley2182	method), 251
method), 397	start_scan() (pymeasure.instruments.srs.SR830
start_buffer() (pymed	
sure.instruments.keithley.Keithley2400	start_sequence() (pymea- sure.instruments.advantest.advantestR624X.AdvantestR624X
method), 349	sure.insirumenis.aavaniesi.aavaniesiK024A,AavanieSiK024A

	method), 120		property), 336
start_s	sequence() (pymed	a- status	(pymeasure.instruments.keithley.Keithley2281S
	sure.instruments.danfysik.Danfysik8500		property), 339
	method), 251	status	(pymeasure.instruments.keithley.Keithley2306
start_s	sequence() (pymed		property), 343
	sure.instruments.rohdeschwarz.hmp.HMP404 method), 544	0 status	(pymeasure.instruments.keithley.Keithley2450 property), 357
start_s	sequence_program() (pymed		(pymeasure.instruments.keithley.Keithley2510
	sure.instruments.advantest.advantestR624X.A		
	method), 121	status	(pymeasure.instruments.keithley.Keithley2600
start_s	step_sweep() (pymed		property), 364
	sure.instruments.agilent.Agilent8257D	status	(pymeasure.instruments.keithley.Keithley2700
ctartur	method), 151 5() (pymeasure.experiment.procedure.Procedu	re status	property), 367 (pymeasure.instruments.keithley.Keithley2750
Star tup	method), 66	re status	property), 370
startur	o() (pymeasure.experiment.procedure.Unknowi	nProstantias	(pymeasure.instruments.keithley.Keithley4200
ocar cap	method), 66	ii roomaaa	property), 405
state	(pymeasure.instruments.aculight.argos.Argo	os status	(pymeasure.instruments.keithley.Keithley6221
	property), 116		property), 376
state	(pymeasure.instruments.ami.AMI430 property 223	), status	(pymeasure.instruments.keithley.Keithley6517B property), 382
status		MUstatus(	pymeasure.instruments.keithley.KeithleyDAQ6510
	property), 190	`	property), 403
status		700Astatus(	pymeasure.instruments.keysight.Keysight81160A
	property), 229		property), 438
status	(pymeasure.instruments.danfysik.Danfysik850 property), 251	00 status(	pymeasure.instruments.keysight.KeysightE36312A property), 419
status		202 <b>56</b> tatus (	pymeasure.instruments.keysight.KeysightE3631A
	property), 255		property), 428
status	(py measure. instruments. fwbell. FWB ell 508)	80 status(	pymeasure.instruments.lecroy.LeCroyT3DSO1204
	property), 259		property), 466
status		<i>xin</i> status(	pymeasure.instruments.mksinst.mks974b.MKS974B
	property), 108	3 <b>.</b>	property), 476
status	(pymeasure.instruments.heidenhain.ND28	37 status(	pymeasure.instruments.mksinst.mksinst.RelayChannel
a+a+	property), 260	.)	property), 472
	(pymeasure.instruments.hp.HP6632A property 318		(pymeasure.instruments.parker.ParkerGV6 property), 506
status	(pymeasure.instruments.hp.HP8116A property 273	), status	(pymeasure.instruments.proterial.ROD4 property), 510
status	(pymeasure.instruments.hp.hp856Xx.HP856X attribute), 276	Xx status	(pymeasure.instruments.ptw.ptwUNIDOS property), 517
status	(pymeasure.instruments.hp.HPLegacyInstrume	nt status(	fy measure. in struments. siglent technologies. SDS 1072CML
	property), 316		property), 549
status	(pymeasure.instruments.Instrument property 105	), status(	pymeasure.instruments.signalrecovery.DSP7225 property), 556
status	(pymeasure.instruments.ipgphotonics.yar.YA property), 322	R status(	pymeasure.instruments.signalrecovery.DSP7265 property), 563
status	(pymeasure.instruments.keithley.Keithley200	00 status(	pymeasure.instruments.spellmanhv.SpellmanXRV
	property), 328		property), 566
status	(pymeasure.instruments.keithley.Keithley218 property), 397	32 status	(pymeasure.instruments.srs.SR510 property), 573
status	(pymeasure.instruments.keithley.Keithley220 property), 334	00 status	(pymeasure.instruments.srs.SR830 property), 578
status	(pymeasure.instruments.keithley.Keithley2260	B status	(pymeasure.instruments.tcpowerconversion.CXN

property), 588 status (pymeasure.instruments.tdk.tdk_gen40_38.TDK_G		
property), 593	method), 544	
status (pymeasure.instruments.tdk.tdk_gen80_65.TDK_G property), 597	e <b>ss80</b> p6&urrent_up() (pymea- sure.instruments.rohdeschwarz.hmp.HMP4040	
${\tt status} (py measure. instruments. teledyne. Teledyne T3AFG$	method), 544	
property), 601	step_points(pymeasure.instruments.agilent.Agilent8257D	
status (pymeasure.instruments.texio.TexioPSW360L30	property), 151	
property), 624	step_position (pymea-	
status (pymeasure.instruments.velleman.VellemanK8090 property), 636	sure.instruments.anaheimautomation.DPSeriesMotorControloproperty), 226	ler
status() (pymeasure.instruments.keithley.Keithley2400		
method), 350	sure.instruments.rohdeschwarz.hmp.HMP4040	
status_byte_register (pymea-	method), 544	
sure. instruments. advantest. advantest R 624 X.		
property), 125	sure.instruments.rohdeschwarz.hmp.HMP4040	
status_desc (pymeasure.instruments.hp.HP3437A at-	method), 544	
tribute), 267	stepd (pymeasure.instruments.attocube.anc300.Axis	
status_desc (pymeasure.instruments.hp.HP3478A at-	property), 248	
tribute), 270	<pre>stepEnabled() (pymeasure.display.inputs.IntegerInput</pre>	
status_desc (pymeasure.instruments.hp.HP6632A at-	method), 78	
tribute), 318	stepEnabled() (pymea-	
status_desc (pymeasure.instruments.hp.HPLegacyInstruattribute), 316	78	
status_hex(pymeasure.instruments.danfysik.Danfysik850		
<pre>property), 251 status_info_shown</pre>	sure.instruments.anaheimautomation.DPSeriesMotorControlomethod), 226	ler
sure.instruments.rohdeschwarz.sfm.SFM		
property), 535	property), 248	
status_preset() (pymea-		
sure.instruments.rohdeschwarz.sfm.SFM method), 535	sure.instruments.hp.hp856Xx.SweepCoupleMode attribute), 303	
status_reg(pymeasure.instruments.rohdeschwarz.sfm.SF property), 535	Mtop (pymeasure.instruments.agilent.agilent4156.VARX property), 171	
StatusCode (class in pymea- sure.instruments.keithley.keithley4200), 406	STOP (pymeasure.instruments.agilent.agilentB1500.StaircaseSweepPost attribute), 197	tOi
StatusRegister (class in pymea- sure.instruments.hp.hp856Xx), 303	stop() (pymeasure.experiment.listeners.Recorder method), 65	
std_current (pymeasure.instruments.keithley.Keithley240 property), 350	O@top() (pymeasure.instruments.advantest.advantestR624X.AdvantestR6 method), 120	624
	5@top() (pymeasure.instruments.advantest.advantestR624X.SMUChann method), 137	ıel
std_resistance (pymea-	stop() (pymeasure.instruments.agilent.agilent4156.Agilent4156	
sure.instruments.keithley.Keithley2400 prop-	method), 168	
erty), 350	stop() (pymeasure.instruments.anaheimautomation.DPSeriesMotorCo	ont
std_resistance (pymea-	method), 226	
sure.instruments.keithley.Keithley2450 property), 357	stop() (pymeasure.instruments.attocube.anc300.Axis method), 248	
	O⊤() (pymeasure.instruments.keysight.KeysightDSOX1102G method), 409	
	5⊤() (pymeasure.instruments.lecroy.LeCroyT3DSO1204 method), 466	
	stop() (pymeasure.instruments.ni.virtualbench.VirtualBench.Function method), 484	Ge

stop O (pymeasure instruments. parker Parker CV6 method), 506 stop O (pymeasure instruments. teledyne Teledyne Oscilloscope method), 604 stop O (pymeasure instruments. teledyne Teledyne Oscilloscope method), 607 stop O (pymeasure instruments. thermotron. Thermotron: 800 method), 626 stop_all O (pymeasure instruments. autocube. anc; 300 ANC; 300 Controller, instruments. danfysik. Danfysik. 8500 method), 284 stop_buffer () (pymeasure instruments. keithley: Keithley2000 method), 285 stop_buffer () (pymeasure instruments. keithley: Keithley2000 method), 375 stop_buffer () (pymeasure instruments. keithley: Keithley2400 method), 375 stop_buffer () (pymeasure instruments. keithley: Keithley2450 method), 367 stop_buffer () (pymeasure instruments. keithley: Keithley2000	stop() (pymeasure.instruments.ni.virtualbench.	VirtualBer		***************************************
method), 506 stop) (pymeasure.instruments.aglient.Aglient8257D stop) (pymeasure.instruments.delient.Aglient8257D method), 604 method), 605 method), 606 stop_all() (pymeasure.instruments.denfysik.Danfysik8500 method), 265 stop_all() (pymeasure.instruments.actocube.anc300.ANC300Controller.instruments.danfysik.Danfysik8500 method), 265 stop_buffer() (pymeasure.instruments.danfysik.Danfysik8500 method), 265 stop_buffer() (pymeasure.instruments.danfysik.Danfysik8500 method), 265 stop_buffer() (pymeasure.instruments.danfysik.Danfysik8500 method), 261 stop_buffer() (pymeasure.instruments.roldeschwarz.hmp.HM94040 method), 544 stop_buffer() (pymeasure.instruments.aglient.Aglien822180 sure.instruments.keithley.Keithley2400 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 376 stop_buffer() (pymeasure.instruments.keithley.Keithley2700 method), 376 stop_buffer() (pymeasure.instruments.keithley.Keithley2621 method), 376 stop_buffer() (pymeasure.instruments.keithley.Keithley2621 method), 376 stop_buffer() (pymeasure.instruments.keithley.Keithley2621 method), 376 stop_frequency (pymeasure.instruments.keithley.Keithley2621 method), 382 stop_buffer() (pymeasure.instruments.keithley.Keithley2700 method), 382 stop_buffer()	method), 488	rkorGV6	- · · · · · · · · · · · · · · · · · · ·	.HP8753E
stop() (pymeasure.instruments.teledyne.TeledyneOscilloscope method), 604 stop() (pymeasure.instruments.thermotron.Thermotron3800 method), 251 stop_all() (pymeasure.instruments.attocube.amc300.ANC300Controller.instruments.danfysik.Danfysik8500 method), 265 stop_buffer() (pymeasure.instruments.keithley.Keithley2000 method), 328 stop_buffer() (pymeasure.instruments.keithley.Keithley2182 method), 328 stop_buffer() (pymeasure.instruments.keithley.Keithley2182 method), 350 stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 350 stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 350 stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley2430 method), 369 stop_buffer() (pymeasure.instruments.keithley.Keithley2430 method), 382 stop_buffer() (pymeasure.instruments.keithley.Keithley2430 method), 382 stop_buffer() (pymeasure.instruments.keithley.Keithley2430 method), 383 stop_buffer() (pymeasure.instruments.keithley.Keithley2430 met		rkerGvo	* * *	gilent8257D
method), 604 stop Q(pymeasure instruments. thermotron. Thermotron. 3800 method), 626 method), 246 stop_buffer() method), 246 stop_buffer() method), 370 stop_buffer() method), 370 stop_buffer() method), 370 stop_buffer() method), 370 stop_buffer() method), 350 stop_buffer() method), 367 stop		neOscillos		g
method), 26 stop_all() (pymeasure:instruments.attocube.anc300ANC300Controller.instruments.danfysik.Danfysik8500 method), 245 stop_buffer() (pymea- sure.instruments.keithley.Keithley2000 method), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2182 method), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2621 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6217 method), 383 stop_buffer() (pymea- sure.instruments.keithley.Keithley6218 method), 383 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 3				k.Danfysik8500
stop_buffer() (pymeasure.instruments.keithley.Keithley2182 method), 397  stop_buffer() (pymeasure.instruments.keithley.Keithley2182 method), 397  stop_buffer() (pymeasure.instruments.keithley.Keithley2182 method), 397  stop_buffer() (pymeasure.instruments.keithley.Keithley2180 method), 350  stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 350  stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 357  stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 367  stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 376  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 367  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 376  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 383  stop_frequency (pymeasure.instruments.keithley.Keithley2210 (pymeasure.instruments.hp.hp856Xx.HP856Xx method), 295  store_onetadata() (pymeasure.instruments.hp.	<pre>stop() (pymeasure.instruments.thermotron.Ther</pre>	motron38	700 method), 251	
method), 246 stop_buffer() (pymea- sure.instruments.keithley.Keithley2000 method), 328 stop_buffer() (pymea- sure.instruments.keithley.Keithley2182 method), 397 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2700 method), 376 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 383 store_method), 284 store_method), 294 store_method), 295 store_entry, 153 stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- sure.instruments.agilent.Agilent8257D prop- sure.instruments.agilent.Agilent8257D prop- sure.instruments.agilent.Agilent8257D prop- sure.instruments.agilent.Agilent8257D prop- sure.instruments.hp.hp856Xx.HP856Xx method), 295 store_entry, 153 stop_frequency (pymea- sure.instruments.hp.hp856Xx.HP856Xx method), 295 store_entry, 153 store_entry, 154 store_entry, 155 store_entry, 154 store_entry, 156 store_entry, 156 store_entry, 157 store_entry, 157 store_entry, 158 store_entry, 158 store_entry, 159 store_entry, 159 store_entry, 159 store_entry, 150 store_en				
sure.instruments.keithley.Keithley2000 method), 328 stop_buffer() (pymea- sure.instruments.keithley.Keithley2182 method), 397 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 376 stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B method), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B method), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B method), 383 stop_frequency (pymea- sure.instruments.advantest.advantestRof24X.Advante	= :	nc300.AN		500
method.), 328 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method.), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2400 method.), 350 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method.), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method.), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method.), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method.), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method.), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method.), 376 stop_buffer() (pymea- sure.instruments.keithley.Keithley621 method.), 376 stop_buffer() (pymea- sure.instruments.keithley.Keithley621 method.), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley617B method.), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B method.), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley621 method.), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley621 method.), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley621 method.), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method.), 382 stop_enety), 183 stop_enety, 183 st		(pymea-	<pre>stop_sequence()</pre>	(рутеа-
sure.instruments.keithley.Keithley2182 method), 397  stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 350  stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 357  stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 367  stop_buffer() (pymeasure.instruments.davantest.advantestR624X.AdvantestR624X property), 126  store_linage() (pymeasure.instruments.anvitsu.AnritsuMS464xB method), 346  store_linage() (pymeasure.instruments.anvitsu.AnritsuMS464xB method), 321  store_linage() (pymeasure.instruments.anvitsu.AnritsuMS464xB method), 348  store_instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymeasure.instruments.anvitsu.AnritsuMS464xB method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymeasure.instruments.keithley.Keithley6517B method), 385  store_onfig (pymeasure.instruments.advantest.advantestR624X.AdvantestR624X.AdvantestR624X.AdvantestR624X.AdvantestR624X.AdvantestR64x.A				HMP4040
method), 397 stop_buffer() (pymeasure.instruments.keithley.Keithley2400 method), 350 stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 357 stop_buffer() (pymeasure.instruments.keithley.Keithley2450 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley2210 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley6221 method), 368 stop_buffer() (pymeasure.instruments.keithley.Keithley65217B method), 382 stop_buffer() (pymeasure.instruments.keithley.Keithley5517B method), 382 stop_buffer() (pymeasure.instruments.keithley.Keithley5517B method), 382 stop_buffer() (pymeasure.instruments.keithley.KeithleyDAQ6510 method), 403 stop_frequency (pymeasure.instruments.advantest.advantestR3767CG.Adsacret/BsiGiffice_command() (pymeasure.instruments.advantest.advantestR624X.AdvantestR624X.advantes	<pre>stop_buffer()</pre>	(pymea-	<pre>stop_step_sweep()</pre>	(pymea-
sure.instruments.keithley.Keithley2400 method), 350 store_Organistruments.keithley.Keithley2450 method), 357 store_buffer() (pymeasure.instruments.dvantest.advantestR624X.AdvantestR624X.				D
stop_buffer() (pymea-sure.instruments.keithley.Keithley2450 method), 357  stop_buffer() (pymea-method), 357  stop_buffer() (pymea-sure.instruments.keithley.Keithley2700 method), 367  stop_buffer() (pymea-sure.instruments.keithley.Keithley2700 method), 367  stop_buffer() (pymea-sure.instruments.keithley.Keithley6221 method), 376  stop_buffer() (pymea-sure.instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymea-sure.instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymea-sure.instruments.keithley.Keithley6517B method), 382  stop_buffer() (pymea-sure.instruments.keithley.Keithley6517B method), 365  stop_buffer() (pymea-sure.instruments.keithley.Keithley6517B method), 373  stop_buffer() (pymea-sure.instruments.keithley.Keithley6517B method), 241  store_measurement (pymea-sure.ement (pymea-sure.ement), 285  stop_frequency (pymea-sure.instruments.hp.hp856Xx.HP856Xx method), 295  stop_frequency (pymea-sure.instruments.agilent.Agilent8257D property), 153  stop_frequency (pymea-sure.instruments.hp.hp856Xx.HP856Xx method), 295  stop_frequency (pymea-sure.instruments.hp.hp856X	stop_buffer()	(рутеа-	StoppableQThread (class in pymeasure.displa	ay.thread),
stop_buffer() (pymea- sure.instruments.keithley.Keithley2450 method), 357 stop_buffer() (pymea- sure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221 method), 376 stop_buffer() (pymea- sure.instruments.keithley.Keithley6317B method), 382 stop_buffer() (pymea- sure.instruments.keithley.Keithley6317B method), 382 stop_buffer() (pymea- sure.instruments.keithley.KeithleyDAQ6510 method), 403 stop_buffer() (pymea- sure.instruments.advantest.advantestR3767CG.AdstorretR8EGMGGe_command() (pymea- sure.instruments.agilent.Agilent4294A prop- erty), 172 stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 153 stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES property), 153 stop_frequency (pymea- sure.instruments.agilent.AgilentE4408B property), 153 stop_frequency (pymea- sure.instruments.agilent.AgilentE4408B property), 159 stop_frequency (pymea- sure.instruments.agilent.agilentE5062A.VNAChamsetbred_readings_count (pymea- sure.instruments.hp.HP34401A property), 159	sure.instruments.keithley.Keithley2400		81	
sure.instruments.keithley.Keithley2450 method), 357 stop_buffer() (pymeasure.instruments.keithley.Keithley2700 method), 367 stop_buffer() (pymeasure.instruments.keithley.Keithley6221 method), 376 stop_buffer() (pymeasure.instruments.keithley.Keithley6221 method), 376 stop_buffer() (pymeasure.instruments.keithley.Keithley6517B method), 382 stop_buffer() (pymeasure.instruments.keithley.KeithleyDAQ6510 method), 403 stop_buffer() (pymeasure.instruments.advantest.adv				method),
stop_buffer() (pymeasure.instruments.keithley.Keithley2700 sure.instruments.keithley.Keithley2700 sure.instruments.keithley.Keithley6221 sure.instruments.keithley.Keithley6221 sure.instruments.keithley.Keithley6221 sure.instruments.keithley.Keithley6218 sure.instruments.keithley.Keithley6517B sure.instruments.keithley.Keithley6517B sure.instruments.keithley.Keithley6517B sure.instruments.keithley.Keithley6517B sure.instruments.keithley.Keithley06210 sure.instruments.keithley.KeithleyDAQ6510 method), 403 store_open() sure.instruments.advantest.advantestR3767CG.Adstarte(B-5004666-command() (pymeasure.instruments.advantest.advantestR3767CG.Adstarte(B-5004666-command()) (pymeasure.instruments.agilent.Agilent4294A property), 172 store_settings() (pymeasure.instruments.agilent.Agilent8257D property), 151 store_settings() (pymeasure.instruments.agilent.Agilent8722ES property), 153 store_offequency (pymeasure.instruments.agilent.AgilentE4408B property), 153 store_offequency (pymeasure.instruments.agilent.AgilentE4408B property), 153 store_instruments.agilent.agilent.AgilentE4408B property), 159 sure.instruments.agilent.agilentE3062A.VNAChanætebred_readings_count (pymeasure.instruments.hp.HP34401A property), 159	=			
stop_buffer() (pymea-				
sure.instruments.keithley.Keithley2700 method), 367 sure.instruments.keithley.Keithley6221 method), 376 stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B sure.instruments.keithley.Keithley6517B stop_buffer() (pymea- method), 382 stop_buffer() (pymea- method), 382 stop_buffer() (pymea- sure.instruments.keithley.KeithleyDAQ6510 method), 403 stop_frequency (pymea- sure.instruments.advantest.advantest.advantest.R767CG.Adstoree(R8EGGGGGe_command() (pymea- sure.instruments.agilent.Agilent4294A prop- erty), 117 stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES property), 153 stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES property), 153 stop_frequency (pymea- sure.instruments.agilent.Agilent8408B property), 153 stop_frequency (pymea- sure.instruments.agilent.AgilentE4408B property), 153 stop_frequency (pymea- sure.instruments.hp.hp856Xx.HP856Xx method), 295 store_leading (pymea- sure.instruments.hp.HP34401A property), store_leading (pymea- sure.instruments.hp.HP34401A property), store_leading (pymea- sure.instruments.hp.HP34401A property), store_leading (pymea- sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property),		,		R624X.AdvantestR624X
stop_buffer() (pymeasure.instruments.keithley.Keithley6221 store_measurement (pymeasure.instruments.keithley.Keithley6211 store_measurement (pymeasure.instruments.keithley.Keithley6517B sure.instruments.keithley.Keithley6517B store_measure.instruments.keithley.Keithley6517B sure.instruments.keithley.KeithleyDAQ6510 sure.instruments.keithley.KeithleyDAQ6510 sure.instruments.advantest.advantestR3767CG.AdstantestR5456Xx method), 295 sure.instruments.advantest.advantestR3767CG.AdstantestR5456Xx method), 295 sure.instruments.advantest.advantestR3767CG.AdstantestR5456Xx method), 295 sure.instruments.agilent.Agilent4294A property), 117 sure.instruments.agilent.Agilent4294A property), 151 stop_frequency (pymeasure.instruments.agilent.Agilent8257D property), 151 store_sure.instruments.agilent.Agilent8257D property), 153 sure.instruments.agilent.Agilent8408B property), 153 sure.instruments.agilent.AgilentE4408B sure.instruments.agilent.AgilentE4408B sure.instruments.agilent.AgilentE4408B sure.instruments.agilent.AgilentE5062A.VNAChanstored_readings_count (pymeasure.instruments.agilent.AgilentE5062A.VNAChanstored_readings_count (pymeasure.instruments.hp.HP34401A property), property), 159	=			,
sure.instruments.keithley.Keithley6221 stree_measurement sure.display.widgets.fileinput_widget.FileInputWidget sure.display.widgets.fileinput_widget.FileInputWidget sure.instruments.keithley.Keithley6517B stree_metadata() (pymeasure.instruments.keithley.Keithley0517B sure.instruments.keithley.KeithleyDAQ6510 sure.instruments.keithley.KeithleyDAQ6510 sure.instruments.advantest.advantest.advantest.advantest.advantest.advantest.hp.hp856Xx.HP856Xx method), 403 sure.instruments.advantest.advantes			<u>-</u>	
stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B store_metadata() (pymea- method), 382 store_metadata() (pymea- sure.instruments.keithley.KeithleyDAQ6510 sure.instruments.keithley.KeithleyDAQ6510 method), 403 store_open() (pymea- sure.instruments.advantest.advantestR3767CG.AdstatestR85f36f3CGe_command() (pymea- sure.instruments.advantest.advantestR3767CG.AdstatestR85f36f3CGe_command() (pymea- sure.instruments.agilent.Agilent4294A prop- erty), 172 store_settings() (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 151 store_settings() (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 151 store_settings() (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 153 store_open() (pymea- sure.instruments.kuhneelectronic.Kusg245_250A store_settings() (pymea- sure.instruments.kuhneelectronic.Kusg245_250A store_short() (pymea- sure.instruments.hp.hp856Xx.HP856Xx method), 295 store_thru() (pymea- sure.instruments.hp.hp856Xx.HP856Xx method), 295 stored_reading (pymea- sure.instruments.hp.HP34401A property), store_readings_count (pymea- sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property),				
stop_buffer() (pymea- sure.instruments.keithley.Keithley6517B stop_buffer() (pymea- method), 382 store_method), 403 store_instruments.keithley.KeithleyDAQ6510 stop_frequency (pymea- property), 117 stop_frequency (pymea- sure.instruments.agilent.Agilent4294A prop- erty), 172 stop_frequency (pymea- sure.instruments.agilent.Agilent4294A prop- erty), 151 stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 151 stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES property), 153 store_method), 295 sure.instruments.kuhneelectronic.Kusg245_250A method), 121 store_settings() (pymea- sure.instruments.kuhneelectronic.Kusg245_250A method), 443 store_short() (pymea- sure.instruments.agilent.Agilent8722ES sure.instruments.agilent.Agilent8722ES sure.instruments.agilent.AgilentE4408B store_frequency (pymea- sure.instruments.agilent.AgilentE4408B store_frequency (pymea- sure.instruments.agilent.AgilentE4408B store_reading (pymea- sure.instruments.hp.HP34401A property), stop_frequency (pymea- sure.instruments.agilent.agilentE5062A.VNAChanstebred_readings_count (pymea- property), 159 sure.instruments.hp.HP34401A property),				4 *
sure.instruments.keithley.Keithley6517B store_metadata() (pymea-method), 382 sure.experiment.results.Results method), stop_buffer() (pymea-sure.instruments.keithley.KeithleyDAQ6510 store_open() (pymea-sure.instruments.advantest.advantestR3767CG.AdstoreuR55fxfxfae_command() (pymea-sure.instruments.advantest.advantestR3767CG.AdstoreuR55fxfxfae_command() (pymea-sure.instruments.agilent.Agilent4294A propserty), 172 stop_frequency (pymea-sure.instruments.agilent.Agilent8257D property), 151 stop_frequency (pymea-sure.instruments.agilent.Agilent8722ES property), 153 store_store_thuc() (pymea-sure.instruments.agilent.AgilentE4408B property), 153 store_frequency (pymea-sure.instruments.agilent.AgilentE4408B property), 153 sure.instruments.hp.hp856Xx.HP856Xx method), 295 store_thuc() (pymea-sure.instruments.agilent.AgilentE4408B store_frequency (pymea-sure.instruments.agilent.AgilentE4408B store_frequency (pymea-sure.instruments.agilent.agilentE5062A.VNAChanstored_readings_count (pymea-sure.instruments.hp.HP34401A property), 159				FileInputWidget
stop_buffer() (pymea- sure.instruments.keithley.KeithleyDAQ6510 store_open() (pymea- method), 403 sure.instruments.hp.hp856Xx.HP856Xx  stop_frequency (pymea- property), 117 sure.instruments.advantest.advantestR3767CG.AdstauretR85QFGFGe_command() (pymea- sure.instruments.advantest.advantestR624X.AdvantestR624X.  sure.instruments.agilent.Agilent4294A prop- erty), 172 sure.instruments.kuhneelectronic.Kusg245_250A  sure.instruments.agilent.Agilent8257D prop- erty), 151 stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 153 store_short() (pymea- sure.instruments.agilent.Agilent8722ES sure.instruments.agilent.Agilent8408B store_thru() (pymea- property), 153 sure.instruments.hp.hp856Xx.HP856Xx sure.instruments.agilent.AgilentE4408B sure.instruments.hp.hp856Xx.HP856Xx sure.instruments.agilent.AgilentE4408B sure.instruments.hp.hp856Xx.HP856Xx sure.instruments.agilent.AgilentE4408B sure.instruments.hp.hp856Xx.HP856Xx sure.instruments.agilent.AgilentE4408B sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property),		4.0		
sure.instruments.keithley.KeithleyDAQ6510 store_open() (pymea-method), 403 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea-sure.instruments.advantest.advantestR3767CG.AdstooretR36ffeffee_command() (pymea-sure.instruments.advantestR624X.AdvantestR624X.		В		
sure.instruments.keithley.KeithleyDAQ6510 store_open() (pymeamethod), 403 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymeasure.instruments.advantest.advantestR3767CG.AdstoretR85f0f6f6e_command() (pymeasure.instruments.advantest.advantestR624X.AdvantestR624X store_troughency (pymeasure.instruments.agilent.Agilent4294A property), 172 store_settings() (pymeasure.instruments.agilent.Agilent8257D property), 151 sure.instruments.agilent.Agilent8257D property), 151 sure.instruments.agilent.Agilent8722ES store_frequency (pymeasure.instruments.agilent.Agilent8722ES store_frequency (pymeasure.instruments.agilent.Agilent8408B stored_reading (pymeasure.instruments.agilent.AgilentE4408B sure.instruments.agilent.agilentE5062A.VNAChansabbred_readings_count (pymeasure.instruments.hp.HP34401A property), 159		(mum a a		metnoa),
stop_frequency (pymea-	=	4 .		(nymaa-
sure.instruments.advantestR3767CG.AdsammetR854ffffGe_command() (pymea-property), 117 sure.instruments.advantestR624X.AdvantestR624X.Stop_frequency (pymea-erty), 172 sure.instruments.kuhneelectronic.Kusg245_250A stop_frequency (pymea-sure.instruments.agilent.Agilent8257D property), 151 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea-sure.instruments.agilent.Agilent8722ES store_thru() (pymea-property), 153 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea-sure.instruments.agilent.AgilentE4408B stored_reading (pymea-property), 153 sure.instruments.hp.HP34401A property), 153 sure.instruments.hp.HP34401A property), 154 sure.instruments.hp.HP34401A property), 155 sure.instruments.agilent.AgilentE5062A.VNAChammebered_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property), 159 sure.instruments.hp.HP34401A property),	method), 403		sure.instruments.hp.hp856Xx.HP856.	
stop_frequency (pymea- sure.instruments.agilent.Agilent4294A prop- erty), 172 sure.instruments.kuhneelectronic.Kusg245_250A stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 151 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES store_thru() (pymea- property), 153 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea- sure.instruments.agilent.AgilentE4408B stored_reading (pymea- property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea- sure.instruments.agilent.agilentE5062A.VNAChansrebred_readings_count (pymea- property), 159 sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property), sure.instruments.hp.HP34401A property),				(mymag
stop_frequency		707CU.A		
sure.instruments.agilent.Agilent4294A property), 172 stop_frequency (pymeasure.instruments.agilent.Agilent8257D property), 151 stop_frequency (pymeasure.instruments.agilent.Agilent8722ES stop_frequency (pymeasure.instruments.agilent.Agilent8722ES stop_frequency (pymeasure.instruments.agilent.Agilent8722ES stop_frequency (pymeasure.instruments.agilent.AgilentE4408B store_drading (pymeasure.instruments.agilent.AgilentE4408B stored_reading (pymeasure.instruments.agilent.AgilentE5062A.VNAChanstebred_readings_count (pymeasure.instruments.hp.HP34401A property), 159 sure.instruments.hp.HP34401A property),		(nymea-		102+21.71avamesi1102+25
stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 151 stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES store_thru() (pymea- property), 153 sure.instruments.agilent.AgilentE4408B stored_reading (pymea- property), 153 sure.instruments.agilent.AgilentE4408B stored_reading (pymea- property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea- property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea- property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea- sure.instruments.agilent.agilentE5062A.VNAChansrebred_readings_count (pymea- property), 159 sure.instruments.hp.HP34401A property),				(pymea-
stop_frequency		r ·r		
sure.instruments.agilent.Agilent8257D prop- erty), 151 sure.instruments.hp.hp856Xx.HP856Xx  stop_frequency (pymea- property), 153 sure.instruments.hp.hp856Xx.HP856Xx  stop_frequency (pymea- property), 153 sure.instruments.hp.hp856Xx.HP856Xx  stop_frequency (pymea- property), 153 sure.instruments.agilent.AgilentE4408B stored_reading (pymea- property), 153 sure.instruments.hp.HP34401A property),  stop_frequency (pymea- property), 153 sure.instruments.hp.HP34401A property),  stop_frequency (pymea- property), 159 sure.instruments.hp.HP34401A property),  sure.instruments.hp.HP34401A property),		(рутеа-		o
stop_frequency (pymea-sure.instruments.agilent.Agilent8722ES store_thru() (pymea-property), 153 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea-sure.instruments.agilent.AgilentE4408B stored_reading (pymea-property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea-sure.instruments.agilent.agilentE5062A.VNAChanstebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),	sure.instruments.agilent.Agilent8257D	prop-	store_short()	(pymea-
sure.instruments.agilent.Agilent8722ES store_thru() (pymea-property), 153 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea-method), 295 sure.instruments.agilent.AgilentE4408B stored_reading (pymea-property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea-265 sure.instruments.agilent.agilentE5062A.VNAChanstebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),	erty), 151		sure.instruments.hp.hp856Xx.HP856	Xx
property), 153 sure.instruments.hp.hp856Xx.HP856Xx stop_frequency (pymea- sure.instruments.agilent.AgilentE4408B stored_reading (pymea- property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea- sure.instruments.agilent.agilentE5062A.VNAChansebred_readings_count (pymea- property), 159 sure.instruments.hp.HP34401A property),		* *		
stop_frequency (pymea-method), 295 sure.instruments.agilent.AgilentE4408B stored_reading (pymea-property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea-265 sure.instruments.agilent.agilentE5062A.VNAChansrebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),		S		4 *
sure.instruments.agilent.AgilentE4408B stored_reading (pymea-property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea-265 sure.instruments.agilent.agilentE5062A.VNAChanstebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),		,		Xx
property), 153 sure.instruments.hp.HP34401A property), stop_frequency (pymea- 265 sure.instruments.agilent.agilentE5062A.VNAChansrebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),		* *		
stop_frequency (pymea- 265 sure.instruments.agilent.agilentE5062A.VNAChansebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),		3	_	= :
sure.instruments.agilent.agilentE5062A.VNAChanssebred_readings_count (pymea-property), 159 sure.instruments.hp.HP34401A property),		(mumaa	-	property),
property), 159 sure.instruments.hp.HP34401A property),				(nymea-
	ŭ ŭ	1. v 1 v1 1C11U	_	* *
stop_frequency (pymea- 265		(рутеа-	265	r ~r//
sure.instruments.hp.hp856Xx.HP856Xx at-StringInput (class in pymeasure.display.inputs), 78			StringInput (class in pymeasure.display.inpu	uts), 78

<pre>strip_chart_dat1 (pymeasure.instrument     property), 584</pre>	s.srs.SR860		pymeasure.instrumen erty), 412	ts.keysight.KeysightN7776C
<pre>strip_chart_dat2 (pymeasure.instrument</pre>	s.srs.SR860		pymeasure.instrumen erty), 644	ts.yokogawa.aq6370series.AQ6370C
<pre>strip_chart_dat3 (pymeasure.instrument</pre>	s.srs.SR860		pymeasure.instrumen erty), 643	ts.yokogawa.aq6370series.AQ6370D
<pre>strip_chart_dat4 (pymeasure.instrument</pre>	s.srs.SR860		pymeasure.instrumen erty), 643	ts.yokogawa.aq6370series.AQ6370E
subsystem_info sure.instruments.rohdeschwarz.sfm.	(pymea- SFM	sweep_speed(	•	ts.yokogawa.aq6370series.AQ6373B
property), 535				ts.yokogawa.aq6370series.AQ6375B
subtract_display_line_from_trace_b( sure.instruments.hp.hp856Xx.HP856		sweep_start	= :	uments.hp.HP8116A
method), 288			erty), 273	I I I I I I I I I I I I I I I I I I I
summary_event				ts.keysight.KeysightN7776C
sure.instruments.keithley.Keithley22	281S		erty), 412	,
property), 340		sweep_status		(pymea-
supply_current (pymeasure.instruments.a	mi.AM1430		nstruments.oxfordinst	ruments.IPS120_10
property), 223	,		erty), 500	,
<pre>sweep_auto_abort()</pre>		sweep_status		(pymea-
sure.instruments.agilent.agilentB15	00.Agilent <b>B</b> 1		nstruments.oxfordinst	ruments.11C303
method), 188	,		erty), 496	1 . 1 . V . 1 . N.777.6
sweep_complete				keysight.KeysightN7776C
sure.instruments.yokogawa.aq6370s	series.AQ037		erty), 412	1 11001164
property), 641	,	_	ymeasure.instruments	s.np.HP8110A prop-
sweep_couple	(pymea-	erty),		4flit
sure.instruments.hp.hp856Xx.HP856 tribute), 293		prope	erty), 497	ts.oxfordinstruments.ITC503
sweep_delay_time				agilent.Agilent8722ES
sure.instruments.advantest.advantes	tR624X.Adva			
property), 120		$sweep\_time(p)$	ymeasure.instruments	agilent.AgilentE4408B
sweep_marker_frequency	(pymea-		erty), 153	
sure.instruments.hp.HP8116A 273	property),		ymeasure.instruments erty), 159	agilent.agilentE5062A.VNAChannel
<pre>sweep_measurement()</pre>				anritsu.anritsuMS464xB.Measuremen
sure.instruments.agilent.Agilent428-	4A		erty), 244	
method), 199			ymeasure.instruments	s.hp.HP8116A prop-
sweep_mode (pymeasure.instruments.anritsu.	anritsuMS46	•		
property), 244				.hp.hp856Xx.HP856Xx
$\verb sweep_mode   (pymeasure.instruments.keysight) $	t.KeysightN7		ute), 293	
property), 412			ymeasure.instruments	s.hp.HP8753E prop-
$\verb sweep_mode   (pymeasure.instruments.yokogav )  $	va.aq6370sei	-		
property), 641				r.rohdeschwarz.fsseries.FSSeries
sweep_output	(pymea-		erty), 540	
sure.instruments.hp.hp856Xx.HP856	6Xx at-	sweep_time_a		(pymea-
tribute), 293				ilentE5062A.VNAChannel
sweep_points	(рутеа-		erty), 159	
sure.instruments.keysight.KeysightN	17776C	sweep_time_i		(рутеа-
property), 412				.aq6370series.AQ6370Series
sweep_rate(pymeasure.instruments.oxfordin	istruments.IF			
property), 500		sweep_timing		(рутеа-
sweep_single	(рутеа-			ilentB1500.AgilentB1500
sure.instruments.hp.hp856Xx.HP856	6Xx at-	metho	pd), 188	
tribute), 293		sweep_twoway		(pymea-

sure.instruments.keysight.KeysightN7776C	property), 517
property), 412	system_number (pymea-
<pre>sweep_type (pymeasure.instruments.agilent.agilentE5062)</pre>	
property), 159	property), 536
sweep_type(pymeasure.instruments.anritsu.anritsuMS46-	
property), 244	sure.instruments.ptw.ptwUNIDOS property),
sweep_wl_start (pymea-	517
sure.instruments.keysight.KeysightN7776C	system_setup (pymea-
property), 412	sure.instruments.keysight.KeysightDSOX1102G
sweep_wl_stop (pymea-	property), 409
sure.instruments.keysight.KeysightN7776C	system_status_code (pymea-
property), 412	sure.instruments.siglenttechnologies.siglent_spdbase.SPDBase
	property), 546
sure.instruments.hp.hp856Xx), 303	system_temp(pymeasure.instruments.toptica.ibeamsmart.IBeamSmart
SweepMode (class in pymea-	property), 635
sure.instruments.advantest.advantestR624X),	system_time(pymeasure.instruments.keithley.KeithleyDMM6500
139	property), 392
SweepMode (class in pymea-	system_voltages (pymea-
sure.instruments.agilent.agilentB1500), 196	sure. instruments. spellmanhv. Spellman XRV
SweepOut (class in pymeasure.instruments.hp.hp856Xx),	property), 566
303	system_voltages (pymea-
SwissArmyFake (class in pymeasure.instruments.fakes),	sure. instruments. spellmanhv. spellman XRV. Un scaled Data
107	property), 567
switch_enabled (pymea-	SystemStatusCode (class in pymea-
sure.instruments.mksinst.mks974b.MKS974B	sure.instruments.siglenttechnologies.siglent_spdbase),
property), 476	547
switch_heater_enabled (pymea-	_
sure.instruments.oxfordinstruments.IPS120_10	
property), 500	Table (class in pymeasure.display.widgets.table_widget),
switch_heater_status (pymea-	90
sure.instruments.oxfordinstruments.IPS120_10	TableWidget (class in pymea-
property), 501	sure.display.widgets.table_widget), 90
switch_off(pymeasure.instruments.velleman.VellemanKo	0000
property), 636	SUBWidget (class in pymea- sure.display.widgets.tab_widget), 87
switch_on (pymeasure.instruments.velleman.VellemanK80	
property), 636	
	erty), 267
	target_current (pymeasure.instruments.ami.AMI430
	property), 223
SwitchHeaterError (class in pymea-	target_field (pymeasure.instruments.ami.AMI430
sure.instruments.oxfordinstruments.ips120_10),	property), 224
501	target_voltage (pymea-
sync_enabled (pymea-	sure.instruments.oxfordinstruments.ITC503
$sure. instruments. rigol. rigol\_dg 800. Voltage Channel and the property of $	
property), 525	target_voltage_table (pymea-
sync_sequence() (pymea-	sure.instruments.oxfordinstruments.ITC503
sure.instruments.danfysik.Danfysik8500	property), 497
method), 251	TC038 (class in pymeasure.instruments.hcp), 260
<pre>synchronous_sweep_source() (pymea-</pre>	TC038D (class in pymeasure.instruments.hcp), 261
sure.instruments.agilent.agilentB1500.SMU	TDK_Gen40_38 (class in pymea-
method), 191	sure.instruments.tdk.tdk_gen40_38), 589
system_current (pymea-	TDK_Gen80_65 (class in pymea-
sure.instruments.temptronic.ATS525 property),	sure.instruments.tdk.tdk_gen80_65), 593
621	TDS2000 (class in pymeasure.instruments.tektronix), 598
<pre>system_info (pymeasure.instruments.ptw.ptwUNIDOS</pre>	

tec (pymeasure.instruments.srs.ldc500series.LDC5 attribute), 568		property), 571 ture (pymeasure.instruments.tcpowerce	onversion CXN
	утеа-	property), 588	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
sure.instruments.thorlabs.ThorlabsPro800		ture (pymeasure.instruments.temptroni	c.ATSBase
property), 627		property), 618	011102000
TEDStatus (pymeasure.instruments.thorlabs.Thorlabs.	absPro8@@mpera	* * *:	on.Thermotron3800
property), 627	<b>.</b>	property), 626	
	<i>ymea</i> - tempera		(pymea-
sure.instruments.teledyne), 610	, <b>.</b>	sure.instruments.oxfordinstruments.ITC	
	ymea-	property), 497	
sure.instruments.teledyne.teledyneMAUI).			(pymea-
612		sure.instruments.oxfordinstruments.ITC	
	утеа-	property), 497	
sure.instruments.teledyne), 602	tempera		(pymea-
TeledyneOscilloscopeChannel (class in py	_	sure.instruments.oxfordinstruments.ITC	
sure.instruments.teledyne.teledyne_oscillo		property), 497	
607		ture_celsius	(pymea-
	утеа-	sure.instruments.lakeshore.LakeShore2	
sure.instruments.teledyne), 599	, mea	property), 446	.11
temp (pymeasure.instruments.toptica.ibeamsmart.II	R <i>eamSma</i> temnera		(рутеа-
property), 635	<i>reamsma</i> bamper a	sure.instruments.temptronic.ATSBase	prop-
temperature (pymeasure.instruments.aculight.arg.	os Argos	erty), 618	prop
property), 117		ture_digits	(рутеа-
temperature (pymeasure.instruments.agilent.Agile	_	sure.instruments.keithley.Keithley2000	4 -
property), 164	11134430A	erty), 328	ргор-
temperature (pymeasure.instruments.agilent.Agile	ntR208 <del>1</del> omnors		(рутеа-
property), 211	mb2905cempera	sure.instruments.keithley.KeithleyDMN	
temperature (pymeasure.instruments.fluke.Fluke	07341	property), 392	10300
property), 256			(рутеа-
temperature (pymeasure.instruments.hcp.TC038	_	sure.instruments.oxfordinstruments.ITC	
erty), 261	ргор-	property), 497	.505
temperature (pymeasure.instruments.hcp.TC	1038D tempera		(рутеа-
property), 261	030D tempera	sure.instruments.temptronic.ATSBase	prop-
* * **	e var VA P	erty), 618	ргор-
temperature (pymeasure.instruments.ipgphotonics property), 322			(mymag
temperature (pymeasure.instruments.keithley.Keit		sure.instruments.lakeshore.LakeShore2	(pymea- 011
	niey2000		.11
property), 328	161 au 2103 amm ama	property), 446	(
temperature (pymeasure.instruments.keithley.Keit	mey2101empera		(pymea-
property), 397	11 2510	sure.instruments.srs.ldc500series.LDC	500Series1EC
temperature (pymeasure.instruments.keithley.Keit		property), 571	(
property), 361	_	ture_kelvin	(pymea-
temperature (pymeasure.instruments.keithley.Keit	hleyDMM6500	sure.instruments.lakeshore.LakeShore2	211
property), 392	: W 1245-25	property), 446	(
temperature (pymeasure.instruments.kuhneelectro	nic.Kusg <b>.emp</b> ezca		(рутеа-
property), 443	0741 MWC074D	sure.instruments.temptronic.ATSBase	prop-
temperature (pymeasure.instruments.mksinst.mks)		erty), 618	/
property), 476	_	ture_limit_air_high	(pymea-
temperature (pymeasure.instruments.oxfordinstrum	ments.mercuryitc	•	prop-
property), 503	MENTOR	erty), 619	,
temperature (pymeasure.instruments.ptw.ptwDIA)	<i>MENTOR</i> empera		(рутеа-
property), 513	a 11 *****	sure.instruments.temptronic.ATSBase	prop-
temperature (pymeasure.instruments.spellmanhv.S	•	erty), 619	,
property), 566	-	ture_limits	(pymea-
temperature (pymeasure instruments srs ldc500se	ries.LDC500Seri	e <b>sut</b> et instruments srs.ldc500series LDC	500SeriesTEC

property), 572		property), 392	
	рутеа-	temperature_seed (pymea-	
sure.instruments.srs.ldc500series.LDC50			
property), 572		erty), 322	
_	рутеа-	temperature_sensor (pymea-	
sure.instruments.keithley.Keithley2000	prop-	sure.instruments.agilent.AgilentB2985 prop-	
erty), 328	r ·r	erty), 211	
	рутеа-	temperature_sensor (pymea-	
sure.instruments.keithley.Keithley2182	prop-	sure.instruments.lakeshore.LakeShore211	
erty), 397	F	property), 446	
	рутеа-	temperature_setpoint (pymea-	
sure.instruments.keithley.KeithleyDMM6	. •	sure.instruments.aculight.argos.Argos prop-	
property), 392		erty), 117	
	nvmea-	temperature_setpoint (pymea-	
		82Channebure.instruments.keithley.Keithley2510 prop-	
property), 399		erty), 361	
	nvmea-	temperature_setpoint (pymea-	
- *		82Channekure.instruments.oxfordinstruments.ITC503	
property), 399		property), 497	
	рутеа-	temperature_setpoint (pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.srs.ldc500series.LDC500Serie	
erty), 361	P · · · P	property), 572	~
	рутеа-	temperature_setpoint (pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.temptronic.ATSBase prop-	
erty), 361	P · · · P	erty), 619	
	рутеа-	temperature_setpoint_window (pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.temptronic.ATSBase prop-	
erty), 361	1 1	erty), 619	
	рутеа-	temperature_simulated_reference (pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.keithley.Keithley2182 prop-	
erty), 361	1 1	erty), 397	
	рутеа-	temperature_soak_time (pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.temptronic.ATSBase prop-	
erty), 361	1 1	erty), 619	
	рутеа-	temperature_unit (pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.agilent.AgilentB2985 prop-	
erty), 361	1 1	erty), 211	
	рутеа-	TemperatureSensor (class in pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.oxfordinstruments.mercuryitc)	,
erty), 361		502	
temperature_protection_range (/	рутеа-	TemperatureStatusCode (class in pymea-	
sure.instruments.keithley.Keithley2510	prop-	sure.instruments.temptronic.temptronic_base),	
erty), 361		620	
temperature_reference ()	рутеа-	$\verb template  (pymeasure. instruments. siglent technologies. SI$	OS1072CM1
sure.instruments.keithley.Keithley2000	prop-	property), 549	
erty), 328		terminals_used (pymea-	
temperature_reference_junction (	рутеа-	sure.instruments.hp.HP34401A property),	
sure.instruments.keithley.Keithley2182	prop-	265	
erty), 397		terminals_used (pymea-	
	рутеа-	sure. instruments. keithley. Keithley DMM 6500	
sure. instruments. keithley. Keithley DMM 6		property), 392	
property), 392		test_method() (pymeasure.generator.Generator	•
	рутеа-	method), 61	
sure.instruments.keithley.KeithleyDMM6	5500	test_property_getter() (pymea-	

	sure.generator.Generator method),	61	sure.insi	truments.signalrecovery.DS	SP7225
test_pro	perty_setter()	(pymea-	property	y), 556	
	sure.generator.Generator method),	61	time_constant		(pymea-
test_pro	perty_setter_batch()	(pymea-	sure.inst	truments.signalrecovery.DS	SP7265
	sure.generator.Generator method),	61	property	y), 563	
TexioPSW	360L30 (class in pymeasure.instrui	nents.texio),	time_constant	(pymeasure.instrumen	ts.srs.SR510
	622		property		
text_ena	bled	(pymea-	time_constant	(pymeasure.instrumen	ats.srs.SR830
	sure.instruments.keithley.Keithley2	700 prop-	property		
	erty), 367	1 1	time_constant	(pymeasure.instrumen	ats.srs.SR860
textFrom	• •	(pymea-	property		
	sure.display.inputs.ScientificInput	method),	time_division	**	(рутеа-
	78	,,		truments.siglenttechnologie	* *
tfi (pvm	easure.instruments.ptw.ptwUNIDO	S property).	property		
	518	- F - F - 5///			t.agilentB1500.AgilentB1500
thermoco	uple	(рутеа-	property	_	8
	sure.instruments.keithley.Keithley2				KeysightDSOX1102G
	erty), 397	P. P	property		,
thermome	* ·	(рутеа-	1 1	sure.instruments.lecroy.Le	CrovT3DSO1204
	sure.instruments.srs.ldc500series.L				
	property), 572			asure.instruments.srs.SR86	60 property).
_	ter_type	(рутеа-	584		r r vy
	sure.instruments.srs.ldc500series.L		<i>TE</i> Cmebase(pvmea	sure.instruments.teledvne.	TeledvneOscilloscope
	property), 572		property		
Thermome		рутеа-	timebase_hor_m		(рутеа-
	sure.instruments.srs.ldc500series),			truments.lecroy.LeCroyT31	
Thermotr		рутеа-	property		
	sure.instruments.thermotron), 625	1 0	timebase_hor_p		(рутеа-
	on3800.Thermotron3800Mode	(class in		truments.lecroy.LeCroyT3L	
	pymeasure.instruments.thermotron)	*	property		
_	measure.instruments.ametek.Amete			,,	(рутеа-
	erty), 221			truments.keysight.Keysight.	4.5
	measure.instruments.srs.SR830 pro	pperty), 578	property		- 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	measure.instruments.srs.SR860 pro		timebase_offse		(рутеа-
	s (pymeasure.instruments.inficon.s				
	property), 321	1	property		- 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
_	PM100USB (class in	рутеа-	timebase_offse		(рутеа-
	sure.instruments.thorlabs), 626	r J		truments.lecroy.LeCroyT3L	* *
Thorlabs		рутеа-	property		
	sure.instruments.thorlabs), 627	Fymu	timebase_offse		(рутеа-
	d (pymeasure.instruments.hp.hp850	6Xx.HP856Xx		truments.teledyne.Teledyne	4.7
	attribute), 276		property	•	
	neasure.instruments.agilent.agilentl	31500.ADCM			(рутеа-
	attribute), 195	3100011120111		truments.keysight.Keysight.	
time	(pymeasure.instruments.fakes.Swi	ssArmvFake	property		- 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	property), 107		timebase_scale		(рутеа-
	neasure.instruments.redpitaya.redpi	itava scni.Red			
	property), 524	.c., csepree	property		220111020
_	pymeasure.instruments.rohdeschwa	arz.sfm.SFM			(рутеа-
	property), 536			truments.lecroy.LeCroyT31	
time_con	* * *	(рутеа-	property	· · · · · · · · · · · · · · · · · · ·	
	sure.instruments.ametek.Ametek727		timebase_scale		(рутеа-
	erty), 221	F - ~ F		truments.teledyne.Teledyne	
time con		(pymea-	property		

timebas	se_setup()	(рутеа-		sure.instruments.agilent.agilentE	5062A.VNAChannel
	sure.instruments.keysight.KeysightDS	OX1102G		property), 160	
	method), 410		trace_m	arker	(рутеа-
timebas	se_setup()	(pymea-		sure.instruments.anritsu.AnritsuM	<i>IS9710C</i>
	sure.instruments.lecroy.LeCroyT3DS0	01204		property), 232	
	method), 466		trace_m	arker_center	(рутеа-
timebas	se_setup()	(pymea-		sure.instruments.anritsu.AnritsuM	<i>IS9710C</i>
	sure. instruments. teledyne. Teledyne Os	cilloscope		property), 232	
	method), 604		trace_m	${\sf ode}(py measure. instruments. rohde$	eschwarz.fsseries.FSSeries
title	(pymeasure.instruments.agilent.Agi	lent4294A		property), 540	
	property), 172			g (pymeasure.instruments.ni.virtua	albench.VirtualBench.PowerSup
to_dict	t() (pymeasure.instruments.agilent.agi	lentB1500.9			
	static method), 193		trackin	g_adjust_coarse	(рутеа-
total_c	cycle_count	(рутеа-		sure.instruments.hp.HP8560A	property),
	sure.instruments.temptronic.ATSBase	prop-		297	
_	erty), 619		trackin	g_adjust_fine	(рутеа-
total_s	so_far (pymeasure.instruments.racal.F property), 520	Racal1992		sure.instruments.hp.HP8560A 298	property),
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63	70 <b>c</b> ackin	g_enabled	(рутеа-
	attribute), 644			sure.instruments.keysight.Keysigh	ntE3631A
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63		property), 428	
	attribute), 643		train_m	_	(рутеа-
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63	70E	sure.instruments.oxfordinstrumen	ts.IPS120_10
	attribute), 642			method), 501	
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63	<i>708<b>ani</b>st</i> e		(рутеа-
	attribute), 639			sure.instruments.yokogawa.aq637	70series.AQ6370Series
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63		property), 641	
	attribute), 644			r_sequence()	(рутеа-
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63		sure.instruments.rohdeschwarz.hr	np.HMP4040
ED 4 (	attribute), 645			method), 544	
TRA (pyn	neasure.instruments.yokogawa.aq6370s	eries.AQ63	/bransla		(pymea-
TD A	attribute), 646	. 100	75 D	sure.display.widgets.table_widget	.PandasModelBase
IKA (pyn	measure.instruments.yokogawa.aq6370s	eries.AQ03		method), 89	(
Т	attribute), 646			te_to_global()	(pymea-
Trace	(class in sure.instruments.anritsu.anritsuMS46	<i>pymea-</i> (4 <i>xB</i> ),		sure.display.widgets.table_widget method), 89	
_ ,	245	CTT \ 201		te_to_global()	(pymea-
	class in pymeasure.instruments.hp.hp85			sure.display.widgets.table_widget	.PandasModelByRow
trace()	(pymeasure.instruments.agilent.Agile	ntE4408B		method), 90	
	method), 153	D.276		te_to_local()	(pymea-
trace_	1 (pymeasure.instruments.advantest.adv property), 117	antestR3/6	/CG.Adva	m <b>thod),</b> 89 method), 89	.PandasModelBase
trace_a	a_minus_b_enabled	(pymea-	transla	te_to_local()	(рутеа-
	sure.instruments.hp.hp856Xx.HP8562 tribute), 289	Xx at-		sure.display.widgets.table_widget method), 89	.PandasModelByColumn
trace_a	a_minus_b_plus_dl_enabled	(pymea-	transla	te_to_local()	(рутеа-
	sure.instruments.hp.hp856Xx.HP8562 tribute), 288	Xx at-		sure.display.widgets.table_widget method), 90	.PandasModelByRow
trace_c	data_format	(рутеа-	TRB (pym	easure.instruments.yokogawa.aq6.	370series.AQ6370C
	sure.instruments.hp.hp856Xx.HP8562		**	attribute), 644	<del></del>
	tribute), 287		TRB (pym	easure.instruments.yokogawa.aq6.	370series.AQ6370D
trace_c	df() (pymeasure.instruments.agilent.Ag	gilentE4408		attribute), 643	
	method), 153		TRB (pym	easure.instruments.yokogawa.aq6.	370series.AQ6370E
trace_f	format	(pymea-		attribute), 642	

- TRB (pymeasure.instruments.yokogawa.aq6370series.AQ637**USE (jey**measure.instruments.yokogawa.aq6370series.AQ6375 attribute), 639 attribute), 646
- TRB (pymeasure.instruments.yokogawa.aq6370series.AQ637BRE (pymeasure.instruments.yokogawa.aq6370series.AQ6375B attribute), 644 attribute), 647
- TRB (pymeasure.instruments.yokogawa.aq6370series.AQ637BBF (pymeasure.instruments.yokogawa.aq6370series.AQ6370C attribute), 645 attribute), 644
- TRB (pymeasure.instruments.yokogawa.aq6370series.AQ637bRF (pymeasure.instruments.yokogawa.aq6370series.AQ6370D attribute), 646 attribute), 643
- TRB (pymeasure.instruments.yokogawa.aq6370series.AQ637**DRF** (pymeasure.instruments.yokogawa.aq6370series.AQ6370E attribute), 646 attribute), 642
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ6370**KF** (pymeasure.instruments.yokogawa.aq6370series.AQ6370Series attribute), 644 attribute), 640
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ637**DRF** (pymeasure.instruments.yokogawa.aq6370series.AQ6373 attribute), 643
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ6370ÆF (pymeasure.instruments.yokogawa.aq6370series.AQ6373B attribute), 642 attribute), 645
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ637**0SE (jey**measure.instruments.yokogawa.aq6370series.AQ6375 attribute), 640 attribute), 646
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ637BRF (pymeasure.instruments.yokogawa.aq6370series.AQ6375B attribute), 644 attribute), 647
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ637BKG (pymeasure.instruments.yokogawa.aq6370series.AQ6370C attribute), 645 attribute), 644
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ637DRG (pymeasure.instruments.yokogawa.aq6370series.AQ6370D attribute), 646 attribute), 643
- TRC (pymeasure.instruments.yokogawa.aq6370series.AQ637**DRG** (pymeasure.instruments.yokogawa.aq6370series.AQ6370E attribute), 647 attribute), 643
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ637**DKG** (pymeasure.instruments.yokogawa.aq6370series.AQ6370Series attribute), 644 attribute), 640
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ637**DRS** (pymeasure.instruments.yokogawa.aq6370series.AQ6373 attribute), 643
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ637**DRG** (pymeasure.instruments.yokogawa.aq6370series.AQ6373B attribute), 642 attribute), 645
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ637**DNG (jey**measure.instruments.yokogawa.aq6370series.AQ6375 attribute), 640 attribute), 646
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ637BRG (pymeasure.instruments.yokogawa.aq6370series.AQ6375B attribute), 645 attribute), 647
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ6378biad() (pymeasure.instruments.keithley.Keithley2400 attribute), 645 method), 350
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ6376riad() (pymeasure.instruments.keithley.Keithley2450 attribute), 646 method), 358
- TRD (pymeasure.instruments.yokogawa.aq6370series.AQ6376bbiad() (pymeasure.instruments.keithley.Keithley2700 attribute), 647 method), 367
- TRE (pymeasure.instruments.yokogawa.aq6370series.AQ6370fiad() (pymeasure.instruments.keithley.Keithley6221 attribute), 644 method), 376
- TRE (pymeasure.instruments.yokogawa.aq6370series.AQ6370Ei angle (pymeasure.instruments.rigol.rigol\_dg800.VoltageChannel attribute), 643 property), 525
- TRE (pymeasure.instruments.yokogawa.aq6370series.AQ6370Figger (pymeasure.instruments.hp.HP3437A propatribute), 642 erty), 267
- TRE (pymeasure.instruments.yokogawa.aq6370series.AQ6370sizgiger (pymeasure.instruments.hp.HP3478A propatribute), 640 erty), 270
- TRE (pymeasure.instruments.yokogawa.aq6370series.AQ637BRIGGER (pymeasure.instruments.hp.hp856Xx.StatusRegister attribute), 645 attribute), 303
- TRE (pymeasure.instruments.yokogawa.aq6370series.AQ6378fi gger (pymeasure.instruments.lecroy.LeCroyT3DSO1204 attribute), 645 property), 466

trigger (pymeasure.instruments.siglenttechnologies.SDS1 attribute), 548	0772iGy@r_acquisition_output_enabled sure.instruments.agilent.AgilentB2981	(pymea- prop-
trigger (pymeasure.instruments.teledyne.TeledyneMAUI	erty), 205	(mym a a
property), 611 trigger (pymeasure.instruments.teledyne.TeledyneOscillo property), 605		(pymea- prop-
trigger() (pymeasure.instruments.activetechnologies.AW		(рутеа-
method), 114 trigger() (pymeasure.instruments.advantest.advantestR6	sure.instruments.agilent.AgilentB2981	prop-
method), 119	· · · · · · · · · · · · · · · · · · ·	(рутеа-
trigger() (pymeasure.instruments.advantest.advantestR6 method), 130	24X.SMUCharanastruments.agilent.AgilentB2981 erty), 205	prop-
trigger() (pymeasure.instruments.agilent.Agilent33220A method), 174	sure.instruments.agilent.AgilentB2981	(pymea- prop-
trigger() (pymeasure.instruments.agilent.Agilent33500	erty), 206	(22222.0.2
method), 177 trigger() (pymeasure.instruments.agilent.Agilent4284A method), 199	trigger_all_bypass_once_enabled sure.instruments.agilent.AgilentB2981 erty), 206	(pymea- prop-
trigger() (pymeasure.instruments.agilent.AgilentE5062A		(рутеа-
method), 157 trigger() (pymeasure.instruments.andeenhagerling.AH2.	sure.instruments.agilent.AgilentB2981 500A erty), 206	prop-
method), 227		(рутеа-
<pre>trigger() (pymeasure.instruments.andeenhagerling.AH2 method), 229</pre>	700A sure.instruments.agilent.AgilentB2981 erty), 206	prop-
trigger() (pymeasure.instruments.anritsu.AnritsuMS464.		(рутеа-
method), 241 trigger() (pymeasure.instruments.keithley.Keithley2182	sure.instruments.agilent.AgilentB2981 erty), 206	prop-
method), 397	• •	(рутеа-
trigger() (pymeasure.instruments.keithley.Keithley2400 method), 350	sure.instruments.agilent.AgilentB2981 erty), 206	prop-
trigger() (pymeasure.instruments.keithley.Keithley2450 method), 358	trigger_all_output_signal sure.instruments.agilent.AgilentB2981	(pymea- prop-
trigger() (pymeasure.instruments.keithley.Keithley6221	erty), 206	ргор-
method), 376		(рутеа-
trigger() (pymeasure.instruments.keithley.Keithley65171 method), 382	erty), 206	prop-
trigger() (pymeasure.instruments.keysight.Keysight8116		(рутеа-
method), 438 trigger() (pymeasure.instruments.philips.PM6669	sure.instruments.agilent.AgilentB2981 erty), 207	prop-
method), 507		(рутеа-
trigger() (pymeasure.instruments.yokogawa.aq6370serie method), 641	erty), 207	prop-
trigger_acquisition_bypass_once_enabled (pymeasure.instruments.agilent.AgilentB2981 property), 204		(pymea- roperty),
trigger_acquisition_count (pymea- sure.instruments.agilent.AgilentB2981 prop- erty), 204	trigger_bus() sure.instruments.agilent.AgilentE5062A method), 157	(pymea- A
trigger_acquisition_delay (pymea-	= =	(pymea-
sure.instruments.agilent.AgilentB2981 property), 205	sure.instruments.agilent.agilentE5062A property), 160	A.VNAChannel
trigger_acquisition_is_idle (pymea-	trigger_continuous	(рутеа-
sure.instruments.agilent.AgilentB2981 property), 205	sure.instruments.hp.HP8753E pi 309	roperty),

	_continuous() sure.instruments.anritsu.AnritsuMS464 method), 241	(pymea- 4xB		method), 350 _immediately() sure.instruments.keithley.Keithley622	(pymea- 21
	_continuous_enabled sure.instruments.agilent.Agilent4284A erty), 199	(pymea- prop-		method), 376 _immediately() sure.instruments.keithley.Keithley651	(pymea-
trigger_	count (pymeasure.instruments.hp.H.	P34401A	+niaaan	method), 382	
	property), 266	(222220 0 0		_in (pymeasure.instruments.keysight.	Keysigmin///OC
trigger_	count sure.instruments.keithley.Keithley2000	(pymea-		property), 412	(mymag
	erty), 329			sure. instruments. a gilent. A gilent 4284	(pymea- A
trigger_		(рутеа-		method), 199	,
	sure.instruments.keithley.Keithley2182	prop-	trigger.		(pymea-
	erty), 397			sure.instruments.agilent.agilentE506	2A.VNAChannel
trigger_		(рутеа-		method), 160	,
	sure.instruments.keithley.Keithley2400	prop-	trigger		(рутеа-
	erty), 350			sure.instruments.advantest.advantest	R624X.SMUChannel
trigger_		(рутеа-		property), 128	
	sure.instruments.keysight.keysight8116	60A.Keysig	e/nt&ilbog@An		(pymea-
	property), 441			sure. instruments. siglent technologies.	siglent_sds1072cml.Trigger
	coupling	(рутеа-		property), 550	
	sure.instruments.siglenttechnologies.si	glent_sds1	0722irgdeFr	iglæ <b>v Eh</b> annel	(pymea-
	property), 550			$sure.instruments.teledyne.teledyne\_o$	scilloscope. Teledyne Oscillo
trigger_	_coupling	(pymea-		property), 610	
	sure.instruments.teledyne.teledyne_osc	illoscope.	Tathe il grgæ00.	<u>s</u> di <b>elwe</b> d@peChannel	(pymea-
	property), 609			sure.instruments.lecroy.lecroyT3DSC	D1204.LeCroyT3DSO1204C
trigger_	_delay	(pymea-		property), 470	
	sure.instruments.agilent.Agilent4284A	prop-	trigger	_level_a	(pymea-
	erty), 199			sure.instruments.racal.Racal1992	property),
trigger_	delay (pymeasure.instruments.hp.Hi	P34401A		520	
	property), 266		trigger	_level_b	(рутеа-
trigger_		(pymea-		sure.instruments.racal.Racal1992	property),
	sure.instruments.keithley.Keithley2000	prop-		520	
	erty), 329		trigger	_link_function_enabled	(рутеа-
trigger_	• .	(рутеа-		sure.instruments.advantest.advantest.	* -
	sure.instruments.keithley.Keithley2182			property), 126	
	erty), 397	1 1		_mode (pymeasure.instruments.h	p.HP437B
trigger_		(рутеа-		property), 315	
	sure.instruments.keithley.Keithley2400		trigger		(рутеа-
	erty), 350	1 1	55	sure.instruments.hp.hp856Xx.HP856	* *
	_delay() (pymeasure.instruments.hp.	HP437B		tribute), 275	
	method), 315		trigger		(рутеа-
trigger_		HP8753E		sure.instruments.keysight.keysight81	* *
	property), 309	11 0,002		property), 441	
	_immediate()	(pymea-	trigger		(рутеа-
	sure.instruments.agilent.Agilent4284A	футей	crrgger.	sure.instruments.lecroy.LeCroyT3DS	
	method), 199			property), 467	01207
	_immediate()	(рутеа-	trigger		(рутеа-
	sure.instruments.hp.HP437B method),	4 .	crigger.	sure.instruments.siglenttechnologies.	
	_immediately()	(pymea-		property), 550	5.5.cm_50510/20111.1118ger
	sure.instruments.keithley.Keithley2182		trigger		(рутеа-
	method), 397		cr rgger.	_mode sure.instruments.teledyne.TeledyneO.	
	_immediately()	(nymea			σειπονεορε
	sure.instruments.keithley.Keithley2400	(pymea-		property), 605 _on_bus()	(pymag
	ын с.нын итеть.кентеу. <b>х</b> ентеу2400		cr ryyer.	_oii_bus()	(рутеа-

sure.instruments.keithley.	.Keithley2182		method), 392		
method), 398		trigger	_slope (pymeasure.instruments.h <sub>l</sub>	p.HP8116A	
trigger_on_bus()	(рутеа-		property), 274		
sure.instruments.keithley.	.Keithley2400	trigger		(рутеа-	
method), 350			sure.instruments.siglenttechnologies	s.siglent_sds1072cml.Trigge	?r(
trigger_on_bus()	(рутеа-		property), 550		
sure.instruments.keithley.	.Keithley6221	trigger		(рутеа-	
method), 376			sure.instruments.teledyne.teledyne_e	os cillos cope. Tele dyne Oscill	los
trigger_on_bus()	(pymea-		property), 610		
sure.instruments.keithley.	.Keithley6517B	trigger	_source	(рутеа-	
method), 382			sure.instruments.activetechnologies.	AWG401x_AWG	
trigger_on_external()	(рутеа-		property), 114		
sure.instruments.keithley.	.Keithley2400	trigger	_source	(рутеа-	
<i>method</i> ), 350			sure.instruments.agilent.Agilent3322	20A	
trigger_on_external()	(рутеа-		property), 174		
sure.instruments.keithley.	Keithley6221	trigger	_source	(рутеа-	
method), 376			sure.instruments.agilent.Agilent3350	00 prop-	
trigger_out( <i>pymeasure.instrume</i>	ents.keysight.KeysightN7	7776C			
property), 412			_source	(рутеа-	
	(рутеа-		sure.instruments.agilent.Agilent4284	4A prop-	
sure.instruments.advante	= -			1 1	
method), 122			_source	(рутеа-	
trigger_output_timing	(рутеа-		sure.instruments.agilent.AgilentE49	* *	
sure.instruments.advante				r · r	
property), 127			'_source	(рутеа-	
	(рутеа-		sure.instruments.agilent.AgilentE50	* *	
sure.instruments.yokogav	4.7		property), 157		
method), 639		triaaer	_source	(рутеа-	
trigger_select	(рутеа-	33	sure.instruments.anritsu.AnritsuMS-	* *	
sure.instruments.lecroy.L			property), 241		
property), 467		triager	_source	(рутеа-	
trigger_select	(рутеа-		sure.instruments.hp.HP34401A	property),	
sure.instruments.teledyne			266	property),	
property), 605		trinner	_source	(рутеа-	
trigger_setup	(рутеа-		sure.instruments.keysight.Keysight8	* *	
sure.instruments.siglentte				1100/1	
property), 550	ennologies.sigieni_sus1	trigger	201	(рутеа-	
trigger_setup()	(рутеа-	crigger	sure.instruments.agilent.Agilent3322	* *	
sure.instruments.lecroy.L			property), 174	20/1	
method), 468	EC10y13D3O1204	triagor	c_sweep()	(рутеа-	
	(рутеа-	trigger	sure.instruments.hp.hp856Xx.HP856	* *	
trigger_setup()	* *				
sure.instruments.teledyne method), 606	reteayneOsciiloscope	+ ni aaar	method), 293	od (mymag	
**	(mym ag	trigger	_transient_bypass_once_enabl sure.instruments.agilent.AgilentB29		
trigger_single()  sure.instruments.agilent.2	(pymea-		<u> </u>	85 prop-	
<u> </u>	AgueniE3002A	+	erty), 211	(	
method), 157	(22,44, 0.2)	trigger	_transient_count	(pymea-	
trigger_single()	(pymea-		sure.instruments.agilent.AgilentB29	85 prop-	
sure.instruments.anritsu.	ARTUSUWI S404XB	+	erty), 211	(2004)	
method), 241	,	crigger	_transient_delay	(pymea-	
trigger_single_autozero()	(pymea-		sure.instruments.agilent.AgilentB29	85 prop-	
sure.instruments.hp.HP3	4401A method),	+	erty), 211	(2004)	
266	,	crigger	_transient_is_idle	(pymea-	
trigger_single_autozero()	(pymea-		sure.instruments.agilent.AgilentB29	85 prop-	
sure.instruments.keithley.	. кентеуыммоэн		erty), 211		

trigger_transient_output_enabled (pymea- sure.instruments.agilent.AgilentB2985 prop- erty), 212	TV_standard(pymeasure.instruments.rohdeschwarz.sfm.SFM property), 527
trigger_transient_output_signal (pymea-	U
sure.instruments.agilent.AgilentB2985 property), 212	unblank_front() (pymeasure.instruments.srs.SR570 method), 574
trigger_transient_source (pymea-	under_voltage (pymea-
sure.instruments.agilent.AgilentB2985 property), 212	sure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38 property), 593
trigger_transient_source_lan_id (pymea-	under_voltage (pymea-
sure.instruments.agilent.AgilentB2985 property), 212	sure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65 property), 597
trigger_transient_timer (pymea-	Uniform (pymeasure.instruments.hp.hp856Xx.WindowType
sure.instruments.agilent.AgilentB2985 prop-	attribute), 304
erty), 212	unique_filename() (in module pymea-
TriggerChannel (class in pymea-	sure.experiment.results), 73
549	1977 (pymeasure.instruments.fluke.Fluke7341 property), 256
triggered_caplossvolt() (pymea- sure.instruments.andeenhagerling.AH2500A	unit (pymeasure.instruments.lakeshore.LakeShore421 property), 453
method), 227	unit (pymeasure.instruments.lakeshore.LakeShore425
triggered_caplossvolt() (pymea- sure.instruments.andeenhagerling.AH2700A	property), 454
method), 229	unit (pymeasure.instruments.lecroy.lecroyT3DSO1204.LeCroyT3DSO1204property), 470
TriggerInputType (class in pymea-	unit (pymeasure.instruments.mksinst.mks937b.MKS937B
sure.instruments.advantest.advantestR624X),	property), 474
140	unit (pymeasure.instruments.mksinst.mks974b.MKS974B
TriggerMode (class in pymea-	property), 476
sure.instruments.hp.hp856Xx), 304	$unit ({\it pymeasure.instruments.tektronix.afg3152c.AFG3152CC} hannel$
TriggerOutputSignalTiming (class in pymea-	property), 599
sure.instruments.advantest.advantestR624X), 142	units (pymeasure.instruments.fwbell.FWBell5080 property), 259
tristate_lines() (pymea-	units (pymeasure,instruments,heidenhain,ND287 prop-
sure. instruments. ni. virtual bench. Virtual Bench. D	igitalInputQ <b>4tpy</b> ,t260
method), 480	units (pymeasure.instruments.newport.esp300.Axis
TS_MB (pymeasure.instruments.oxfordinstruments.Mercury	p. op e. c.y),
attribute), 502	units (pymeasure.instruments.srs.ldc500series.LDC500SeriesPD
ts1570 (in module pymeasure.instruments.santec), 545	property), 570
tune() (pymeasure.instruments.kuhneelectronic.Kusg245 method), 443	* * * * * * * * * * * * * * * * * * * *
tune_capacity (pymea-	sure.experiment.procedure), 66
sure.instruments.tcpowerconversion.CXN	unlock_harmonic_number() (pymea- sure.instruments.hp.HP8561B method), 300
property), 588	unscaled (pymeasure.instruments.spellmanhv.SpellmanXRV
tune_capacity (pymea-	attribute), 564
sure.instruments.tcpowerconversion.tccxn.Preset	ChasestedData (class in pymea-
property), 588	sure.instruments.spellmanhv.spellmanXRV),
tuner (pymeasure.instruments.tcpowerconversion.CXN	567
property), 588	update() (pymeasure.display.curves.Crosshairs
turn_off() (pymeasure.instruments.kuhneelectronic.Kus	
method), 444	update_channels() (pymea-
turn_on() (pymeasure.instruments.kuhneelectronic.Kusg method), 444	
TV_country (pymeasure.instruments.rohdeschwarz.sfm.SI	method), 242
property), 526	'Update_data() (pymea- sure.display.curves.ResultsCurve method),

76				ure.display.inputs.Sci	entificInput
•	(pymea-		ethod), 78	1. 1	
sure.display.widgets.estimator_widget.1 method), 82	Estimator		) (pymeasure ethod), 83	display.widgets.filen:	ame_widget.FilenameValida
<pre>update_frequency_range()</pre>				.display.widgets.sequ	encer_widget.ExpressionVal
method), 244	AD.MCusu			_terminal()	(рутеа-
	(рутеа-				tualBench.DigitalMultimete
sure.experiment.experiment.Experimen			ethod), 481	is.ni.viriudioenen.vir	indiBenen.DigitalMattimete.
method), 64	ı	validate_			(рутеа-
update_parameter() (pymeasure.display.inp	uts.Input			ts.ni.virtualbench.Vir	tualBench.MixedSignalOsci
method), 77			ethod), 488		
	(рутеа-	validate_			(рутеа-
sure.experiment.experiment.Experimen				ts.ni.virtualbench.Vir	tualBench.PowerSupply
method), 64			ethod), 489		
update_status()	(рутеа-	validate_	dmm_functi	on()	(рутеа-
sure.experiment.workers.Worker	nethod),		ıre.instrumen ethod), 481	ts.ni.virtualbench.Vir	tualBench.DigitalMultimete
. —	(pvmea-	validate_			(рутеа-
sure.instruments.anritsu.anritsuMS464	4 .			ts.ni.virtualbench.Vir	4.7
method), 245			ethod), 480		0 1 1
	(pymea-	validate_			(рутеа-
sure.display.widgets.sequencer_widget.	ComboBo	xDelegate si	ıre.instrumen	ts.ni.virtualbench.Vir	tualBench.DigitalMultimete
method), 85			atic method),		-
updateEditorGeometry()	(рутеа-	validate_	trigger_in	stance()	(рутеа-
sure.display.widgets.sequencer_widget. method), 86	LineEditI		ire.instrumen atic method),		tualBench.MixedSignalOsci
use_absolute_position()	(рутеа-	valueFrom			(рутеа-
sure.instruments.parker.ParkerGV6 1 506	nethod),	sı 73		puts.ScientificInput	method),
use_external_source	(рутеа-	values()(	pymeasure.in	struments.common_b	oase.CommonBase
sure.instruments.rohdeschwarz.sfm.Sou	nd_Chan	nel m	ethod), 102		
property), 537			pymeasure.in	struments.hp.HPLego	acyInstrument
	(рутеа-	m	ethod), 316		
sure.instruments.keithley.Keithley2400 method), 350			pymeasure.in ethod), 438	struments.keysight.K	eysight81160A
use_front_terminals()	(рутеа-	values()(	pymeasure.in	struments.keysight.K	eysightE36312A
sure.instruments.keithley.Keithley2450		m	ethod), 420		
method), 358		values()(	pymeasure.in	struments.keysight.K	eysightE3631A
use_rear_terminals()	(рутеа-		ethod), 428		
sure.instruments.keithley.Keithley2400 method), 350			pymeasure.in ethod), 468	struments.lecroy.LeC	SroyT3DSO1204
use_rear_terminals()	(рутеа-	values()(	pymeasure.in	struments.rohdeschw	arz.sfm.Sound_Channel
sure.instruments.keithley.Keithley2450		m	ethod), 538		
method), 358		values()(	pymeasure.in	struments.tcpowerco	nversion.CXN
use_relative_position()	(рутеа-	m	ethod), 588		
sure.instruments.parker.ParkerGV6 1 506	nethod),		pymeasure.in ethod), 588	struments.tcpowerco	nversion.tccxn.PresetChanne
user_tag(pymeasure.instruments.mksinst.mks9	74b.MKS9			instruments.proteria	l.rod4.ROD4Channel
property), 476			roperty), 511	•	
		valve_sca	ling		(pymea-
V				ts.oxfordinstruments.	ITC503
V (pymeasure.instruments.hp.hp856Xx.Ampliti	ıdeUnits	-	roperty), 497		
attribute), 300		VAR1	(class	in	рутеа-

vare.instruments.agilent.agile		Video (pymeasure.instruments.hp.hp856Xx.Trigg attribute), 304	gerMode
sure.instruments.agilent.a	1 2		(рутеа-
	in pymea-	sure.instruments.hp.hp856Xx.HP856Xx	4 -
sure.instruments.agilent.a	1 2	tribute), 284	ı cı
_			(рутеа-
sure.instruments.agilent.a		sure.instruments.hp.hp856Xx.HP856Xx	4.
vbs_ask() (pymeasure.instruments	•		
method), 611			(pymea-
<pre>vbs_write() (pymeasure.instrument</pre>	nts.teledyne.TeledyneM	IAUI sure.instruments.rohdeschwarz.fsseries	.FSSeries
method), 612		property), 541	
VectorParameter (class	in pymea-	video_bandwidth_to_resolution_bandwid	th
sure.experiment.paramete	rs), 71	(pymeasure.instruments.hp.hp856Xx.H	P856Xx
VellemanK8090 (class	in pymea-	attribute), 284	
sure.instruments.velleman		55	(pymea-
VellemanK8090Switches (cla		sure.instruments.hp.hp856Xx.HP856Xx	x at-
sure.instruments.velleman	* *	tribute), 284	
velocity (pymeasure.instrumen	nts.parker.ParkerGV6		(pymea-
property), 506		sure.instruments.anritsu.AnritsuMS209	90A
<pre>verify_calibration_data()</pre>	(pymea-	property), 236	
sure.instruments.hp.HP34			(рутеа-
<pre>verify_calibration_entry()</pre>	(pymea-	sure.instruments.hp.hp856Xx.HP856Xx	r
sure.instruments.hp.HP34		method), 286	
version (pymeasure.adapters.Pro	ologixAdapter prop-		рутеа-
erty), 56		sure.instruments.ni.virtualbench), 478	
property), 117	.acungni.argos.Argos	VirtualBench.DigitalInputOutput (class in sure.instruments.ni.virtualbench), 479	ı рутеа-
	tocube.anc300.ANC30	OV introdleBench.DigitalMultimeter (class in	рутеа-
property), 246		sure.instruments.ni.virtualbench), 480	
${\tt version}(py measure. instruments. ku$	thneelectronic.Kusg24.	5_ <b>V256</b> AualBench.FunctionGenerator (class in	рутеа-
property), 444		sure.instruments.ni.virtualbench), 482	
version(pymeasure.instruments.ox	cfordinstruments.IPS12	${\it 20} { t V10}$ tual ${ t Bench.MixedSignal0scilloscope}$ (	
property), 501		pymeasure.instruments.ni.virtualbench	), 484
	cfordinstruments.ITC50	O3VirtualBench.PowerSupply (class in	рутеа-
property), 497		sure.instruments.ni.virtualbench), 488	
version (pymeasure.instruments.p	roterial.ROD4 prop-		рутеа-
erty), 510		sure.instruments.ni.virtualbench), 491	
	hdeschwarz.hmp.HMP	24049AAdapter (class in pymeasure.adapters), 49	
property), 544	II . II 40 30 TD V		(pymea-
= :	k.tak_gen40_38.1DK_	Gen40_38 sure.instruments.agilent.agilentE5062A	A. VNAChanne
property), 593	11-411 00 65 TDV	property), 160	(
version (pymeasure.instruments.td property), 597	к.taк_gen80_03.1 <b>DK</b> _	_	(pymea- M
version (pymeasure.instruments.to	ontica ibaamsmart IRaa	sure.instruments.rohdeschwarz.sfm.SFl umSmart property), 536	VI
property), 635	рисальеатьтанливеа		(рутеа-
version (pymeasure.instruments.ve	elleman VellemanK809		
property), 636	neman. venemanixoox	property), 536	,,
vertical_division	(pymea-		(рутеа-
	2.7	1072cml.Vo <b>lsuge.Gistmun</b> knts.rohdeschwarz.sfm.SFI	
property), 549		property), 536	
vhighest (pymeasure.instruments.c	andeenhagerling.AH25		(рутеа-
property), 227		sure.instruments.rohdeschwarz.sfm.SFI	
vhighest (pymeasure.instruments.c	andeenhagerling.AH27		
property), 229		vision_clamping_average	(pymea-

sure.instruments.rohdeschwarz.sfm.SFM	property), 255
property), 536	voltage (pymeasure.instruments.fakes.SwissArmyFake
vision_clamping_enabled (pymea-	property), 107
sure.instruments.rohdeschwarz.sfm.SFM	voltage (pymeasure.instruments.hp.HP6632A prop-
property), 536	erty), 318
vision_clamping_mode (pymea-	voltage (pymeasure.instruments.keithley.Keithley2000
sure.instruments.rohdeschwarz.sfm.SFM	property), 329
property), 536	voltage (pymeasure.instruments.keithley.Keithley2182
vision_precorrection_enabled (pymea-	
sure.instruments.rohdeschwarz.sfm.SFM	voltage (pymeasure.instruments.keithley.keithley2200.PSChannel
property), 536	property), 334
vision_residual_carrier_level (pymea-	
sure.instruments.rohdeschwarz.sfm.SFM	property), 336
property), 536	voltage (pymeasure.instruments.keithley.Keithley2400
vision_sideband_filter_enabled (pymea-	
sure.instruments.rohdeschwarz.sfm.SFM	voltage (pymeasure.instruments.keithley.Keithley2450
property), 536	property), 358
	voltage (pymeasure.instruments.keithley.Keithley2510
sure.instruments.rohdeschwarz.sfm.SFM	property), 361
property), 536	voltage (pymeasure.instruments.keithley.keithley4200.SMU
visit_tree() (pymea-	* * *
	er <b>TrækMegte(</b> pymeasure.instruments.keithley.Keithley6517B
method), 87	property), 382
	voltage (pymeasure.instruments.keithley.KeithleyDAQ6510
sure.instruments.agilent.agilent4156), 171	property), 403
	voltage (pymeasure.instruments.keithley.KeithleyDMM6500
sure.instruments.agilent.agilentE5062A),	property), 392
157	voltage (pymeasure.instruments.kepco.KepcoBOP3612
VNATrace (class in pymea-	
sure.instruments.agilent.agilentE5062A),	voltage (pymeasure.instruments.keysight.keysightE36312A.VoltageChann
160	property), 421
	voltage (pymeasure.instruments.keysight.keysightE3631A.VoltageChanne
property), 164	property), 429
	leasOptiage(pymeasure.instruments.keysight.KeysightN5767A
attribute), 195	property), 411
	voltage (pymeasure.instruments.oxfordinstruments.mercuryitc.Heater
property), 213	property), 503
	SMIOChaga (pymeasure.instruments.oxfordinstruments.mercuryitc.Temperatu
property), 214	property), 503
	nnedoltage (pymeasure.instruments.ptw.ptwUNIDOS prop-
property), 216	erty), 518
voltage (pymeasure.instruments.aimtti.ld400p.LD400P	
property), 218	property), 544
voltage (pymeasure.instruments.aja.DCXS property),	
220	property), 547
voltage (pymeasure.instruments.ametek.Ametek7270	- ··
property), 222	property), 556
	voltage (pymeasure.instruments.signalrecovery.DSP7265
property), 248	
VOLTAGE Invineasure instruments blanceision RK Precision	property), 563
	on916DBage (pymeasure.instruments.spellmanhv.SpellmanXRV
property), 249	on <b>916DB</b> age (pymeasure.instruments.spellmanhv.SpellmanXRV property), 566
property), 249 voltage (pymeasure.instruments.deltaelektronika.SM704.	on9IdDBage (pymeasure.instruments.spellmanhv.SpellmanXRV property), 566 45Boltage (pymeasure.instruments.spellmanhv.spellmanXRV.UnscaledData
property), 249 voltage (pymeasure.instruments.deltaelektronika.SM704. property), 253	on <b>916DB</b> age (pymeasure.instruments.spellmanhv.SpellmanXRV property), 566

property), 569			sure.instruments.keithley.KeithleyDMM	16500
voltage (pymeasure.instruments.srs.ldc500seri	es.LDC500			
property), 572		_	_ac_relative_enabled	(pymea-
voltage (pymeasure.instruments.tdk.tdk_gen40_property), 593	_38.TDK_	Gen40_38	sure.instruments.keithley.KeithleyDMN property), 393	M6500
voltage(pymeasure.instruments.tdk.tdk_gen80	65.TDK	GeorBCa65	* * * .	(pymea-
property), 597			sure.instruments.agilent.Agilent344502	* *
voltage (pymeasure.instruments.texio.TexioPS)	W360L30		property), 165	
property), 624	7200220	voltane	_amplitude	(рутеа-
voltage_1 (pymeasure.instruments.razorbill.ra.	zorbillRP1		sure.instruments.activetechnologies.AV	* *
property), 522			property), 115	, o rora. Chamacarr o
voltage_2 (pymeasure.instruments.razorbill.ra.	zorbillRP1	' <i>0</i> ∕∕ooltage	_amplitude_max	(pymea-
property), 522			sure.instruments.activetechnologies.AV	VG401x.ChannelAFG
voltage_32v (pymeasure.instruments.kuhneele	ctronic.Ku	sg245_250	Aroperty), 115	
property), 444		voltage	_amplitude_min	(pymea-
voltage_5v(pymeasure.instruments.kuhneelect	tronic.Kus	g245_250A	Asure.instruments.activetechnologies.AV	VG401x.ChannelAFG
property), 444			property), 115	
voltage_ac(pymeasure.instruments.agilent.Ag	ilent34410	Avoltage		(pymea-
property), 161		5	sure.instruments.rohdeschwarz.hmp.H.	
voltage_ac(pymeasure.instruments.agilent.Ag	ilent34450	)A	property), 544	
property), 164			_auto_range	(pymea-
voltage_ac (pymeasure.instruments.hp.H.	P34401A		sure.instruments.agilent.Agilent34450a	
property), 266			property), 165	
voltage_ac_auto_range	(nymea-	voltage	_dc (pymeasure.instruments.agilent.Agi	ilent34410A
sure.instruments.agilent.Agilent34450.		vorcage	property), 161	11011
property), 164		voltage		(рутеа-
voltage_ac_bandwidth	(рутеа-	vortage	sure.instruments.keithley.Keithley2000	4.5
sure.instruments.keithley.Keithley2000			erty), 329	ргор
erty), 329	ргор-	voltage		(рутеа-
voltage_ac_bandwidth	(mymaa	vortage	_urgits sure.instruments.keithley.KeithleyDMN	
	(pymea-			10300
sure.instruments.keithley.KeithleyDMN	M0300	]+	property), 393	(20000000000000000000000000000000000000
property), 392	(******* * **	vortage	_filter_count	(pymea-
voltage_ac_digits	(рутеа-		sure.instruments.keithley.Keithley2450	prop-
sure.instruments.keithley.Keithley2000	prop-	1	erty), 358	,
erty), 329	(	voitage	_filter_type	(pymea-
voltage_ac_digits	(pymea-		sure.instruments.keithley.Keithley2450	prop-
sure.instruments.keithley.KeithleyDMN	10300	<b>.</b>	erty), 358	
property), 392		voltage		(pymea-
voltage_ac_nplc	(рутеа-		sure.instruments.advantest.advantestRe	624X.SMUChannel
sure.instruments.keithley.Keithley2000	prop-	_	method), 131	,
erty), 329		voltage	_fixed_pulsed_sweep()	(pymea-
voltage_ac_range	(рутеа-		sure.instruments.advantest.advantestRe	624X.SMUChannel
sure.instruments.agilent.Agilent34450.	A		method), 131	
property), 165		voltage		(pymea-
voltage_ac_range	(рутеа-		sure.instruments.active technologies. AV	VG401x.ChannelAFG
sure.instruments.keithley.Keithley2000	prop-		property), 115	
erty), 329		voltage	_high	(рутеа-
voltage_ac_range	(рутеа-		sure.instruments.agilent.Agilent33220a	A
sure.instruments.keithley.KeithleyDMM	<i>16500</i>		property), 174	
property), 393		voltage	_high	(pymea-
voltage_ac_reference	(рутеа-		sure. instruments. a gilent. A gilent 33500	prop-
sure.instruments.keithley.Keithley2000	prop-		erty), 177	
erty), 329		voltage		(pymea-
voltage_ac_relative	(pymea-		sure.instruments.agilent.agilent33500.	Agilent33500Channel

	property), 180			erty), 398	
voltage		(рутеа-	voltage.	_nplc	(рутеа-
	sure.instruments.keysight.Keysight811	60A		sure.instruments.keithley.Keithley2400	prop-
	property), 439			erty), 350	• •
	_high_max	(рутеа-		• •	(рутеа-
				Gsure.instruments.keithley.Keithley2450	prop-
	property), 115			erty), 358	1 1
	_high_min	(pymea-	voltage	• •	(рутеа-
3				Gsure.instruments.keithley.Keithley65171	4.5
	property), 115			property), 382	
	_limit (pymeasure.instruments.ami	AMI430			(рутеа-
	property), 224			sure.instruments.keithley.KeithleyDAQ0	
voltage.		(рутеа-		property), 404	
	sure.instruments.keithley.keithley2200.				(рутеа-
	property), 334	- ~ ~		sure.instruments.keithley.KeithleyDMM	
voltage.		(рутеа-		property), 393	
· o _ cago	 sure.instruments.oxfordinstruments.me				(рутеа-
	property), 503			sure.instruments.activetechnologies.AW	A .
voltage.		(рутеа-		property), 116	370134 CHANNELLII 3
vor cage.	sure.instruments.srs.ldc500series.LDC				(рутеа-
	property), 569	2005011051		sure.instruments.keithley.keithley2182	4.5
voltage.		(рутеа-		property), 399	itemmey 21 02 emanner
vor cage.	sure.instruments.srs.ldc500series.LDC				(рутеа-
	property), 572	200501105		sure.instruments.keithley.keithley2182	
voltage.		(рутеа-		property), 399	Acimic y21 02 Chamilet
voi cage.	sure.instruments.yokogawa.Yokogawa(				(рутеа-
	property), 639	35200		sure.instruments.activetechnologies.AW	4.7
		(рутеа-		property), 116	0+01x.Channell 11 0
voi cage.	sure.instruments.keithley.keithley2200.				(рутеа-
	property), 334	1 Senanne		sure.instruments.activetechnologies.AW	4.
	_low(pymeasure.instruments.activetecl	hnologies A			o roix. Chamica ii o
voi cage.	property), 115	inologics.			(рутеа-
voltane	_low(pymeasure.instruments.agilent.A	ailent3322		sure.instruments.keithley.Keithley2450	
	property), 174	guemisszz		erty), 358	ргор
	_low(pymeasure.instruments.agilent.A	ailent3350		* * *	(рутеа-
voi cage.	property), 177	guemissso		_pu13eu_3ou1ee() sure.instruments.advantest.advantestR6	
voltane	_low(pymeasure.instruments.agilent.ag	ailent3350			2+A.SMO Channei
voi cage.	property), 180	Sucrussion			(рутеа-
voltane	_low(pymeasure.instruments.keysight.l	Kevsiaht81	_	_pu13eu_sweep() sure.instruments.advantest.advantestR6	= :
voi cage.	property), 439	ic ysigmor		method), 132	2+A.SMO Channei
voltane	_low_max	(nymea-	voltage.		(рутеа-
voi cage.				Gsure.instruments.eurotest.EurotestHPP.	
	property), 115	10401x.CI		property), 255	120230
anc+Iov	_low_min	(nvmea-			(рутеа-
voi cage.			_	_random_purscu_sweep() Isure.instruments.advantest.advantestR6	4.2
	property), 116	10101x.CI	umitum C	method), 132	2+A.SMO Channei
voltage.	·	(nvmea-	vol+ane		(рутеа-
voi cage			_	_1 andom_sweep() I <b>sann:eh</b> struments.advantest.advantestR6	
	property), 169	guenunea		method), 132	24A.SMOCHannet
voltage		(рутеа-	voltage.		(рутеа-
vor cage	_nprc sure.instruments.keithley.Keithley2000	4.2	voi tage	_1 ange sure.instruments.agilent.Agilent34450A	* *
	erty), 329	ριομ-		property), 165	•
1	0.011, 040			property, 100	
VOITAGE	nnlc	(nymea-			(pymea-
voitage.	_nplc sure.instruments.keithley.Keithley2182	(pymea- prop-	voltage.		(pymea- prop-

erty), 213			property), 255	
voltage_range	(pymea-		_setpoint	(рутеа-
sure.instruments.eurotest.EurotestHPF	2120256		sure. instruments. keithley. keithley 2200	.PSChannel
property), 255			property), 334	
voltage_range			_setpoint	(рутеа-
sure.instruments.keithley.Keithley2000	prop-		sure.instruments.keithley.Keithley2260	$\partial B$
erty), 329			property), 336	
voltage_range			_setpoint	(рутеа-
sure.instruments.keithley.keithley2182. property), 399	.Keithley2		ebure.instruments.keithley.keithley4200 property), 406	.SMU
voltage_range	(рутеа-	voltage.	_setpoint	(рутеа-
sure.instruments.keithley.Keithley2400	prop-		sure.instruments.kepco.KepcoBOP361	2
erty), 351			property), 408	
voltage_range	(рутеа-		_setpoint	(рутеа-
sure.instruments.keithley.Keithley2450 erty), 358	prop-		sure.instruments.keysight.keysightE36. property), 421	312A.VoltageChannel
voltage_range	(рутеа-	voltage.	_setpoint	(рутеа-
sure.instruments.keithley.Keithley6517 property), 382	'B		sure.instruments.keysight.keysightE36.property), 429	31A.VoltageChannel
voltage_range	(рутеа-	voltage.	_setpoint	(рутеа-
sure.instruments.keithley.KeithleyDAQ	6510		sure.instruments.siglenttechnologies.si	iglent_spdbase.SPDChanne
property), 404			property), 547	
voltage_range	(рутеа-	voltage.	_setpoint	(рутеа-
sure.instruments.keithley.KeithleyDMN	16500		sure.instruments.spellmanhv.Spe	XRV
property), 393			property), 566	
voltage_range			_setpoint	(рутеа-
sure.instruments.keysight.KeysightN57 property), 411	'67A		sure.instruments.spellmanhv.spellman.property), 567	XRV.UnscaledData
			_setpoint	(рутеа-
sure.instruments.keithley.keithley2182.	.Keithley2	182Channe	ekure.instruments.tdk.tdk_gen40_38.TD	0K_Gen40_38
property), 399			property), 593	
voltage_reference			_setpoint	(рутеа-
sure.instruments.keithley.Keithley2000 erty), 329			sure.instruments.tdk.tdk_gen80_65.TD property), 597	OK_Gen80_65
voltage_relative			_setpoint	(рутеа-
sure.instruments.keithley.KeithleyDMN property), 393	16500		sure.instruments.texio.TexioPSW360L. property), 624	30
voltage_relative_enabled	(рутеа-			(рутеа-
sure.instruments.keithley.KeithleyDMN			sure.instruments.advantest.advantestR	624X.SMUChannel
property), 393			method), 130	
voltage_resolution	(рутеа-	voltage.	_step	(рутеа-
sure.instruments.agilent.Agilent34450a	A		sure.instruments.rohdeschwarz.hmp.H	MP4040
property), 165			property), 544	
<pre>voltage_set_random_memory()</pre>	(рутеа-	voltage.	_sweep()	(рутеа-
sure.instruments.advantest.advantestRest	624X.SMU	IChannel	sure. instruments. advantest. advantest R	624X.SMUChannel
method), 133			method), 131	
voltage_setpoint		_	_to_max()	(рутеа-
sure.instruments.agilent.agilentE5270.	B.SMUCh		sure.instruments.rohdeschwarz.hmp.H	MP4040
property), 214			method), 544	
voltage_setpoint	(рутеа-	voltage.	_to_min()	(pymea-
sure.instruments.aimtti.aimttiPL.PLCh	annel		sure.instruments.rohdeschwarz.hmp.H	MP4040
property), 216	(		method), 544	(2004)
voltage_setpoint sure.instruments.eurotest.EurotestHPF	(pymea- 2120256	voltage	_unit sure.instruments.activetechnologies.AV	(pymea- NGA01x ChannelAEG
sure.msn unicitis.euroiest.Eurolestiii i	120230		sarcansir umenus. activetectinologies. A)	TOTOIA. CHAIIIIEIAI O

	property), 116			wait_for() (pymeasure.instruments.keithley.Keithley2182
Voltage	*	elass in eysight.keysightE36.	<i>pymea-</i>	method), 398 wait_for() (pymeasure.instruments.keithley.Keithley2200
	420	eysigni.keysigniE50.	312A),	method), 334
Voltage		class in	рутеа-	wait_for() (pymeasure.instruments.keithley.Keithley2260B
	sure.instruments.k	eysight.keysightE36.	31A),	method), 337
3	428			wait_for() (pymeasure.instruments.keithley.Keithley2281S
Voltage		class in	pymea-	method), 340
Voltage		rigol.rigol_dg800), 5 class in	24 pymea-	wait_for() (pymeasure.instruments.keithley.Keithley2306 method), 343
vorcage	*			% of a rail of the contract of
	549	0	0 –	method), 351
Voltage			рутеа-	<pre>wait_for() (pymeasure.instruments.keithley.Keithley2450</pre>
		idvantest.advantestR	624X),	method), 358
VCII	139	:		wait_for() (pymeasure.instruments.keithley.Keithley2510
VSH	(class	in hyracont.smartline_1	pymea- v2)	<pre>method), 361 wait_for() (pymeasure.instruments.keithley.Keithley2600</pre>
	632	nyracom.smariime_	v 2 ),	method), 364
VSM	(class	in	рутеа-	wait_for() (pymeasure.instruments.keithley.Keithley2700
		hyracont.smartline_	v2),	method), 367
	632			wait_for() (pymeasure.instruments.keithley.Keithley2750
VSP	(class	in In an anti-an anti-an a	pymea-	method), 370
	632	hyracont.smartline_	V2),	wait_for() (pymeasure.instruments.keithley.Keithley6221 method), 376
VSR	(class	in	рутеа-	wait_for() (pymeasure.instruments.keithley.Keithley6517B
	,	hyracont.smartline_		method), 382
	632			$wait\_for()$ (pymeasure.instruments.keithley.KeithleyDAQ6510
VSU	(class	in	pymea-	method), 404
	sure.instruments.a	igilent.agilent4156),	1/1	wait_for() (pymeasure.instruments.keysight.Keysight81160A method), 439
W				wait_for() (pymeasure.instruments.keysight.KeysightE36312A
	neasure instruments	s.hp.hp856Xx.Amplit	tudeUnits	method), 420
п фун	attribute), 300	pp0302111.piii	uuc Omns	<pre>wait_for() (pymeasure.instruments.keysight.KeysightE3631A</pre>
wait()		ents.anritsu.Anritsu	MS9710C	method), 428
	method), 232			wait_for() (pymeasure.instruments.lecroy.LeCroyT3DSO1204
wait()		ents.siglenttechnolog	gies.SDS10	072CML method), 469 wait_for() (pymeasure.instruments.proterial.ROD4
i+ f	method), 549	struments.aculight.a	waas Awaas	
walt_10	method), 117	sirumenis.acuiigni.a	rgos.Argos	wait_for() (pymeasure.instruments.signalrecovery.DSP7225
wait_fo	* *	struments.andeenhag	gerling.AH.	2700A method), 556
	method), 229			<pre>wait_for() (pymeasure.instruments.signalrecovery.DSP7265</pre>
wait_fo		struments.attocube.a	nc300.AN	C300Controllethod), 563
• • •	method), 247		.1 1	<pre>wait_for() (pymeasure.instruments.spellmanhv.SpellmanXRV</pre>
wait_fc	or() (pymeasure.in 106	struments.Channel	method),	wait_for() (pymeasure.instruments.tdk.tdk_gen40_38.TDK_Gen40_38
wait fo		struments.common_l	base.Comn	
	method), 103	<u>-</u> c		<pre>wait_for() (pymeasure.instruments.tdk.tdk_gen80_65.TDK_Gen80_65</pre>
wait_fo	or() (pymeasure.ins	struments.fwbell.FW	Bell5080	method), 597
	method), 259			wait_for() (pymeasure.instruments.teledyne.TeledyneT3AFG
wait_fo		asure.instruments.In	strument	<pre>method), 601 wait_for() (pymeasure.instruments.temptronic.ATSBase</pre>
wait fo	method), 105	struments.keithley.Ke	eithlev2000	. T. C10
wart_IC	method), 329	ы ынсніз.кеннісу.К	c y 2000	wait_for()(pymeasure.instruments.texio.TexioPSW360L30
	,, 0-2			method), 624

<pre>wait_for_buffer()</pre>	method), 255
sure.instruments.keithley.Keithley2000	<pre>wait_for_ready()</pre>
method), 329	sure.instruments.danfysik.Danfysik8500
<pre>wait_for_buffer()</pre>	method), 251
sure.instruments.keithley.Keithley2182	<pre>wait_for_settling()</pre>
method), 398	sure.instruments.temptronic.ATSBase method),
<pre>wait_for_buffer()</pre>	619
sure.instruments.keithley.Keithley2400	<pre>wait_for_srq() (pymeasure.adapters.PrologixAdapter</pre>
method), 351	method), 56
	<pre>wait_for_srq() (pymeasure.adapters.VISAAdapter</pre>
sure.instruments.keithley.Keithley2450	method), 50
method), 358	wait_for_stop() (pymea-
wait_for_buffer() (pymea-	sure.instruments.newport.esp300.Axis method),
sure.instruments.keithley.Keithley2700	477
method), 368	wait_for_sweep() (pymea-
wait_for_buffer() (pymea-	sure.instruments.anritsu.AnritsuMS9710C
sure.instruments.keithley.Keithley6221	method), 232
method), 376	wait_for_sweep_complete() (pymea-
<pre>wait_for_buffer()</pre>	sure.instruments.yokogawa.aq6370series.AQ6370Series method), 641
method), 382	wait_for_temperature() (pymea-
wait_for_buffer() (pymea-	sure.instruments.lakeshore.lakeshore_base.LakeShoreTemperatur
sure.instruments.keithley.KeithleyDAQ6510	method), 455
method), 404	wait_for_temperature() (pymea-
wait_for_buffer() (pymea-	sure.instruments.oxfordinstruments.ITC503
sure.instruments.signalrecovery.DSP7225	method), 497
method), 556	wait_for_temperature_stable() (pymea-
wait_for_buffer() (pymea-	sure.instruments.keithley.Keithley2510
sure.instruments.signalrecovery.DSP7265	method), 362
method), 563	<pre>wait_for_temperature_stable()</pre>
<pre>wait_for_buffer() (pymeasure.instruments.srs.SR830</pre>	sure.instruments.srs.ldc500series.LDC500SeriesTEC
method), 579	method), 572
<pre>wait_for_complete()</pre>	<pre>wait_for_trigger()</pre>
sure.instruments.agilent.AgilentE5062A	sure.instruments.agilent.Agilent33220A
method), 157	method), 174
	<pre>wait_for_trigger()</pre>
	NotorControl <b>lere</b> .instruments.agilent.Agilent33500 method),
method), 226	177
wait_for_current() (pymea-	wait_for_trigger() (pymea-
sure.instruments.danfysik.Danfysik8500	sure.instruments.keysight.Keysight81160A
method), 251	method), 439
wait_for_data() (pymea-	wait_time() (pymeasure.instruments.agilent.agilentB1500.AgilentB1500
sure.experiment.experiment.Experiment method), 64	method), 187
**	<pre>wait_to_continue()</pre>
wait_for_holding() (pymea- sure.instruments.ami.AMI430 method), 224	method), 408
wait_for_idle() (pymea-	WaitTimeType (class in pymea-
sure.instruments.oxfordinstruments.IPS120_10	sure.instruments.agilent.agilentB1500), 197
method), 501	wave (pymeasure.instruments.fakes.SwissArmyFake
wait_for_measurement() (pymea-	property), 107
sure.instruments.racal.Racal1992 method),	waveform (pymeasure.instruments.rigol.rigol_dg800.VoltageChannel
520	property), 525
<pre>wait_for_output_voltage_reached() (pymea-</pre>	waveform() (pymeasure.instruments.tektronix.afg3152c.AFG3152CChann
sure.instruments.eurotest.EurotestHPP120256	method), 599

sure.instruments.keithley.Keithley6221	waveform_points (pymea- sure.instruments.teledyne.TeledyneOscilloscope
method), 376	property), 606
	waveform_points_mode (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keysight.KeysightDSOX1102G
erty), 377	property), 410
waveform_arm() (pymea-	waveform_preamble (pymea-
sure.instruments.keithley.Keithley6221	sure.instruments.keysight.KeysightDSOX1102G
method), 377	property), 410
	waveform_preamble (pymea-
sure.instruments.keysight.KeysightDSOX1102G	sure.instruments.lecroy.LeCroyT3DSO1204
property), 410	property), 469
	waveform_preamble (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.teledyne.TeledyneOscilloscope
erty), 377	property), 607
<pre>waveform_duration_set_infinity() (pymea-</pre>	waveform_ranging (pymea-
sure.instruments.keithley.Keithley6221	sure.instruments.keithley.Keithley6221 prop-
method), 377	erty), 377
	waveform_setup (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.siglenttechnologies.SDS1072CML
erty), 377	property), 549
* *	
* *	waveform_source (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keysight.KeysightDSOX1102G
erty), 377	property), 410
	waveform_sparsing (pymea-
sure.instruments.lecroy.LeCroyT3DSO1204	sure.instruments.lecroy.LeCroyT3DSO1204
property), 469	property), 469
<pre>waveform_first_point</pre>	waveform_sparsing (pymea-
sure.instruments.teledyne.TeledyneOscilloscope	sure.instruments.teledyne.TeledyneOscilloscope
property), 606	property), 607
	waveform_start() (pymea-
sure.instruments.keysight.KeysightDSOX1102G	sure.instruments.keithley.Keithley6221
property), 410	method), 377
	waveform_use_phasemarker (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keithley.Keithley6221 prop-
erty), 377	erty), 377
	waveform_volatile (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keysight.keysight81160A.Keysight81160AChann
erty), 377	property), 441
waveform_offset (pymea-	waveforms (pymeasure.instruments.activetechnologies.AWG401x_AWG
sure.instruments.keithley.Keithley6221 prop-	property), 114
erty), 377	waveforms (pymeasure.instruments.keysight.keysight81160A.Keysight8116
•	property), 441
sure.instruments.keithley.Keithley6221 prop-	wavelength(pymeasure.instruments.keysight.KeysightN7776C
erty), 377	property), 412
waveform_phasemarker_phase (pymea-	${\tt wavelength} \ (py measure. instruments. thorlabs. Thorlabs PM 100 USB$
sure.instruments.keithley.Keithley6221 prop-	property), 626
erty), 377	wavelength_automatic_center (pymea-
waveform_points (pymea-	sure.instruments.yokogawa.aq6370series.AQ6370Series
sure.instruments.keysight.KeysightDSOX1102G	property), 642
property), 410	wavelength_center (pymea-
waveform_points (pymea-	sure.instruments.anritsu.AnritsuMS9710C
sure.instruments.lecroy.LeCroyT3DS01204	property), 232
property), 469	wavelength_center (pymea-
property, +U7	wavelength_center (Dymea-

	sure.instruments.yokogawa.aq6370seri	ies.AQ6370	Ostania e ()	(pymeasure.adapters.FakeAdapter method), 59
	property), 642		write()	(pymeasure.adapters.PrologixAdapter method),
waveleng	gth_marker_value	(pymea-		57
	sure.instruments.anritsu.AnritsuMS97	10C	<pre>write()</pre>	(pymeasure.adapters.SerialAdapter method), 52
	property), 232		<pre>write()</pre>	(pymeasure.adapters.VISAAdapter method), 50
waveleng	gth_max	(pymea-	<pre>write()</pre>	(py measure. instruments. advantest. advantest R 624 X. Advantest R
	sure.instruments.thorlabs.ThorlabsPM	100USB		method), 119
	property), 627		write()	(pymeasure.instruments.agilent.agilentB1500.SMU
waveleng	gth_min	(pymea-		method), 189
	sure.instruments.thorlabs.ThorlabsPM property), 627	100USB	write()	(pymeasure.instruments.anaheimautomation.DPSeriesMotorCont method), 226
	gth_span	(pymea-	write()	(pymeasure.instruments.andeenhagerling.AH2700A
	sure.instruments.anritsu.AnritsuMS971			method), 229
	property), 232		write()	(pymeasure.instruments.Channel method), 106
	gth_span	(рутеа-		(pymeasure.instruments.eurotest.EurotestHPP120256
	sure.instruments.yokogawa.aq6370seri			method), 256
	property), 642		write()	
	gth_start	(рутеа-		method), 259
	sure.instruments.anritsu.AnritsuMS971		write()	(pymeasure.instruments.hcp.TC038 method),
	property), 232			261
	gth_start	(nymea-	write()	(pymeasure.instruments.hcp.TC038D method),
-	sure.instruments.yokogawa.aq6370seri			261
	property), 642	cs.11Q0570	write()	
	gth_stop	(рутеа-	wiite()	method), 266
	sure.instruments.anritsu.AnritsuMS971		writa()	(pymeasure.instruments.hp.HP8116A method),
	property), 232	100	wiite()	274
	gth_stop	(mymaa	writa()	(pymeasure.instruments.hp.HPLegacyInstrument
	sure.instruments.yokogawa.aq6370seri			method), 317
	property), 642	es.AQ0570		(pymeasure.instruments.inficon.sqm160.SQM160
		(123,122,00	wiite()	
	gth_temperature	(pymea-	·····: + o ()	method), 321
	· · · · · · · · · · · · · · · · ·	prop-	write()	(pymeasure.instruments.Instrument method), 105
	erty), 322	(	·····: + o ()	
	gth_value_in		write()	(pymeasure.instruments.keithley.Keithley2000
	sure.instruments.anritsu.AnritsuMS971	100		method), 330
	property), 232			(pymeasure.instruments.keithley.Keithley2182
	gths (pymeasure.instruments.anritsu.A	nriisum <b>s</b> 9		method), 398
	property), 232			(pymeasure.instruments.keithley.Keithley2260B
	e (pymeasure.instruments.teledyne.teled	lyne I 3AF C		
	property), 602		write()	(pymeasure.instruments.keithley.Keithley2281S
WindowTy	• =	рутеа-		method), 340
	sure.instruments.hp.hp856Xx), 304	,	write()	**
_	eep_table()	(pymea-		method), 343
	sure.instruments.oxfordinstruments.ITC	2503	write()	4.5
	method), 498			method), 351
wires	(pymeasure.instruments.keithley.Keith	hley2400	write()	The state of the s
	property), 351			method), 359
wires	(pymeasure.instruments.keithley.Keith	hley2450	write()	The state of the s
	property), 359			method), 362
	ing(pymeasure.instruments.keysight.K	eysightN77	7Monocite()	
	property), 412			method), 364
	nfig (pymeasure.instruments.ptw.ptw	UNIDOS	write()	* · · · · · · · · · · · · · · · · · · ·
	property), 518	7.1		method), 368
	class in pymeasure.experiment.workers		write()	* · · · · · · · · · · · · · · · · · · ·
write()	(pymeasure.adapters.Adapter method),	48		method), 370

write()	(pymeasure.instruments.keithley.Keithley6221 method), 377	write() (pymeasure.instruments.velleman.Velleman	nanK8090
write()	(pymeasure.instruments.keithley.Keithley6517B method), 382	write_binary_values() (pymeasure.adapters. method), 48	Adapter
write()	(pymeasure.instruments.keithley.KeithleyDAQ651	<pre>0write_binary_values()</pre>	(рутеа-
	method), 404	sure.adapters.FakeAdapter method), 60	
write()	(pymeasure.instruments.keithley.keithleyDMM650	OG.Siction biological 2000 Cheschel	(рутеа-
	method), 394	sure.adapters.PrologixAdapter n	nethod),
write()	(pymeasure.instruments.keysight.Keysight81160A	57	
	method), 439	<pre>write_binary_values()</pre>	(рутеа-
write()	(pymeasure.instruments.keysight.KeysightE36312.	A sure.adapters.SerialAdapter method), 5	3
	method), 420	<pre>write_binary_values()</pre>	(рутеа-
write()	(pymeasure.instruments.keysight.KeysightE3631A	sure.adapters.VISAAdapter method), 51	
	method), 428	<pre>write_binary_values()</pre>	(рутеа-
write()	(pymeasure.instruments.kuhneelectronic.Kusg245 method), 444	_250A sure.instruments.andeenhagerling.AH2 method), 229	700A
write()	(pymeasure.instruments.lakeshore.LakeShore421	<pre>write_binary_values()</pre>	(рутеа-
	method), 453	sure.instruments.Channel method), 106	
write()	(pymeasure.instruments.lecroy.LeCroyT3DSO120	<pre>#rite_binary_values()</pre>	(рутеа-
	method), 469	sure.instruments.fwbell.FWBell5080 n	nethod),
write()	(pymeasure.instruments.mksinst.mksinst.MKSInst	rument 259	
	method), 471	<pre>write_binary_values()</pre>	(рутеа-
write()	(pymeasure.instruments.ni.virtualbench.VirtualBench.vir	ench.Digita <b>sImpuit(Strtpm</b> ents.Instrument method), 1	05
	method), 480	write_binary_values()	(рутеа-
write()	(pymeasure.instruments.ox for dinstruments.base. Oxfordinstruments.base. Oxf	OxfordInstru <b>smæninBausæ</b> nents.keithley.Keithley2000	
	method), 494	method), 330	
write()	(py measure. instruments. proterial. ROD4)		(рутеа-
	method), 511	sure.instruments.keithley.Keithley2182	
write()	(pymeasure.instruments.racal.Racal1992	method), 398	
	method), 521	•	(рутеа-
	(pymeasure.instruments.signalrecovery.DSP7225 method), 556	sure.instruments.keithley.Keithley2200 method), 334	
write()	(pymeasure.instruments.signalrecovery.DSP7265		(рутеа-
	method), 563	sure.instruments.keithley.Keithley22601	3
write()	(pymeasure.instruments.spellmanhv.SpellmanXRV		
	method), 566		(рутеа-
	(pymeasure.instruments.tcpowerconversion.CXN	sure.instruments.keithley.Keithley22815	<b>S</b>
	method), 588	method), 340	,
write()	(pymeasure.instruments.tdk.tdk_gen40_38.TDK_0		(рутеа-
	method), 593	sure.instruments.keithley.Keithley2306	
write()	(pymeasure.instruments.tdk.tdk_gen80_65.TDK_0		(
··ni+a()	method), 597		(рутеа-
	(pymeasure.instruments.teledyne.TeledyneOscillos method), 607	method), 351	
write()	(pymeasure.instruments.teledyne.TeledyneT3AFG	•	(рутеа-
	method), 601	sure.instruments.keithley.Keithley2450	
write()	(pymeasure.instruments.texio.TexioPSW360L30	method), 359	
	method), 624		(рутеа-
write()	(pymeasure.instruments.thermotron.Thermotron3	· · · · · · · · · · · · · · · · · · ·	
	method), 626	method), 362	,
write()	(pymeasure.instruments.thyracont.smartline_v1.St		(рутеа-
	method), 628	sure.instruments.keithley.Keithley2600	
write()	(pymeasure.instruments.thyracont.smartline_v2.St		(
	method), 631	<pre>write_binary_values()</pre>	(рутеа-

sure.instruments.keithley.Keithley2700 method), 368	)	<pre>method), 53 write_bytes() (pyr</pre>	neasure.adapters.VISAAdapter
write_binary_values()	(nymaa	method), 51	пеизиге.иииріетs. v Ізллииріет
The state of the s	(pymea-		(22,000,000
sure.instruments.keithley.Keithley2750	)	write_bytes()	(pymea-
method), 370	,		.andeenhagerling.AH2700A
<pre>write_binary_values()</pre>	(pymea-	method), 230	
sure.instruments.keithley.Keithley622	<u>l</u>		ymeasure.instruments.Channei
method), 377		method), 106	,
<pre>write_binary_values()</pre>		write_bytes()	(рутеа-
sure.instruments.keithley.Keithley6517 method), 382		259	.fwbell.FWBell5080 method).
<pre>write_binary_values()</pre>	(рутеа-	write_bytes() (pym	neasure.instruments.Instrument
sure.instruments.keithley.KeithleyDAQ	96510	method), 105	
method), $404$		write_bytes()	(рутеа-
write_binary_values()	(рутеа-	sure.instruments.	.keithley.Keithley2000
sure.instruments.keysight.Keysight811	60A	method), 330	
method), 439		<pre>write_bytes()</pre>	(рутеа-
<pre>write_binary_values()</pre>	(pymea-	sure.instruments.	.keithley.Keithley2182
sure.instruments.keysight.KeysightE36	6312A	method), 398	
method), 420		<pre>write_bytes()</pre>	(рутеа-
<pre>write_binary_values()</pre>	(рутеа-	sure.instruments.	.keithley.Keithley2200
sure.instruments.keysight.KeysightE36	531A	method), 334	
method), 428		write_bytes()	(рутеа-
<pre>write_binary_values()</pre>	(рутеа-	-	.keithley.Keithley2260B
sure.instruments.lecroy.LeCroyT3DSC	* *	method), 337	, ,
method), 470		write_bytes()	(pymea-
write_binary_values()	(рутеа-	-	keithley.Keithley2281S
sure.instruments.proterial.ROD4		method), 340	
511	memou),	write_bytes()	(рутеа-
write_binary_values()	(рутеа-		.keithley.Keithley2306
sure.instruments.signalrecovery.DSP7	* *	method), 343	.neumey.neumey2300
method), 557		write_bytes()	(рутеа-
write_binary_values()	(рутеа-		keithley.Keithley2400.
sure.instruments.signalrecovery.DSP7	* *	method), 351	.Ketimey.Ketimey2400
method), 563		write_bytes()	(рутеа-
		-	keithley.Keithley2450.
<pre>write_binary_values()</pre>	V = (pymeu - 1)	38 method) 350	.ketimey.Ketimey2450
			(22,000,000
		<pre>write_bytes()</pre>	(pymea-
write_binary_values()	(pymea-		.keithley.Keithley2510
sure.instruments.tdk.tdk_gen80_65.TL	JK_Gen80_		,
method), 598	,	write_bytes()	(pymea-
<pre>write_binary_values()</pre>	(pymea-		.keithley.Keithley2600
sure.instruments.teledyne.TeledyneT3A	AFG	method), 364	,
method), 601		write_bytes()	(pymea-
write_binary_values()	(рутеа-		.keithley.Keithley2700
sure.instruments.texio.TexioPSW360L	30	method), 368	
method), 624		write_bytes()	(рутеа-
<pre>write_bytes() (pymeasure.adapters.Adapter</pre>	method),	sure.instruments.	.keithley.Keithley2750
48		method), 370	
<pre>write_bytes() (pymeasure.adapters.Fak</pre>	xeAdapter	write_bytes()	(рутеа-
method), 60		sure.instruments.	.keithley.Keithley6221
<pre>write_bytes() (pymeasure.adapters.Prologi</pre>	ixAdapter	method), 378	
method), 57		<pre>write_bytes()</pre>	(pymea-
write_bytes() (pymeasure.adapters.Seria	alAdapter	sure.instruments.	.keithley.Keithley6517B

method), 383		sure.generator.Generator method), 61
write_bytes()	(рутеа-	<pre>write_property_tests()</pre>
sure.instruments.keithley.KeithleyDAgenstruments.keithleyDAgenstruments.keithley.keithleyDAgenstruments.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithley.keithley.keithleyDAgenstruments.keithleyDag	Q6510	sure.generator.Generator method), 61
method), 404		write_setter_test() (pymea-
<pre>write_bytes()</pre>	(pymea-	sure.generator.Generator method), 61
sure.instruments.keysight.Keysight81	160A	<pre>writeAO() (in module pymeasure.instruments.comedi),</pre>
method), 439		110
write_bytes()	(рутеа-	X
sure.instruments.keysight.KeysightE3	6312A	^
method), 420		x (pymeasure.instruments.ametek.Ametek7270 property),
write_bytes()	(рутеа-	222
sure.instruments.keysight.KeysightE3 method), 428	631A	x (pymeasure.instruments.signalrecovery.DSP7225 property), 557
write_bytes()	(рутеа-	x (pymeasure.instruments.signalrecovery.DSP7265 prop-
sure.instruments.lecroy.LeCroyT3DS	01204	erty), 564
method), 470		x (pymeasure.instruments.srs.SR830 property), 579
write_bytes()	(рутеа-	x (pymeasure.instruments.srs.SR860 property), 585
sure.instruments.proterial.ROD4 511	method),	x1 (pymeasure.instruments.ametek.Ametek7270 prop- erty), 222
write_bytes()	(рутеа-	x2 (pymeasure.instruments.ametek.Ametek7270 prop-
sure.instruments.signalrecovery.DSP/	7225	erty), 222
method), 557		x_pointer(pymeasure.instruments.oxfordinstruments.ITC503
write_bytes()	(рутеа-	property), 498
sure.instruments.signal recovery.DSPT	7265	xroll_frequency (pymea-
method), 563		sure.instruments.hp.hp856Xx.HP856Xx at-
write_bytes()	(рутеа-	tribute), 292
sure.instruments.tdk.tdk_gen40_38.Ti method), 593	DK_Gen40_	_ <b>38y</b> (pymeasure.instruments.ametek.Ametek7270 prop- erty), 222
write_bytes()	(рутеа-	xy (pymeasure.instruments.signalrecovery.DSP7225
sure.instruments.tdk.tdk_gen80_65.Th	DK_Gen80	
method), 598		xy (pymeasure.instruments.signalrecovery.DSP7265
write_bytes()	(рутеа-	property), 564
sure. instruments. teledyne. Teledyne T3	AFG	xy (pymeasure.instruments.srs.SR830 property), 579
method), 601		<b>V</b>
write_bytes()	(рутеа-	Υ
sure.instruments.texio.TexioPSW360L method), 624	L30	y (pymeasure.instruments.ametek.Ametek7270 property), 222
<pre>write_calibration_data()</pre>	(рутеа-	y (pymeasure.instruments.signalrecovery.DSP7225 prop-
sure.instruments.hp.HP3478A method	d), 271	erty), 557
<pre>write_composition()</pre>	(рутеа-	y (pymeasure.instruments.signalrecovery.DSP7265 prop-
$sure.instruments.thyracont.smartline\_$	_v2.Smartlii	neV2 erty), 564
method), 631		y (pymeasure.instruments.srs.SR830 property), 579
write_enabled	(рутеа-	y (pymeasure.instruments.srs.SR860 property), 585
sure.instruments.ptw.ptwUNIDOS 518	property),	y1 (pymeasure.instruments.ametek.Ametek7270 prop-
write_file() (pymeasure.generator. method), 61	Generator	erty), 222 y2 (pymeasure.instruments.ametek.Ametek7270 prop-
write_getter_test()	(рутеа-	erty), 222
sure.generator.Generator method), 61	* *	y_pointer(pymeasure.instruments.oxfordinstruments.ITC503
write_init_test() (pymeasure.generator.		property), 498
method), 61	- civer and i	YAR (class in pymeasure.instruments.ipgphotonics.yar),
write_method_test()	(рутеа-	321
sure.generator.Generator method), 61		YAR. Status (class in pymea-
write method tests()	(nvmea-	sure.instruments.ipgphotonics.yar), 321

Yokogawa7651 (class in рутеаsure.instruments.yokogawa), 637 YokogawaGS200 (class рутеаsure.instruments.yokogawa), 639 Z zero() (pymeasure.instruments.ami.AMI430 method), (pymeasure.instruments.hp.HP437B method), zero() 316 (pymeasure.instruments.newport.esp 300. Axiszero() method), 477 zero() (pymeasure.instruments.ptw.ptwUNIDOS method), 518 zero\_corrected (pymeasure.instruments.agilent.AgilentB2981 property), 207 zero\_probe() (pymeasure.instruments.lake shore.Lake Shore 421*method*), 453 zero\_probe() (pymeasure.instruments.lake Shore .Lake Shore 425method), 454 zero\_status (pymeasure.instruments.ptw.ptwUNIDOS property), 518