# **PyMeasure Documentation**

Release 0.9.0

**PyMeasure Developers** 

# **LEARNING PYMEASURE**

1	Introduction       1.1 Instrument ready	3 3
2	Quick start2.1Setting up Python2.2Installing PyMeasure	<b>5</b> 5 5
3	Tutorials 3.1 Connecting to an instrument	<b>7</b> 7 9 16
4	4.1 Adapter base class 4.2 Fake adapter 4.3 Serial adapter 4.4 Prologix adapter 4.5 VISA adapter 4.6 VXI-11 adapter	31 32 33 34 36 38
5	5.1 Experiment class 5.2 Listener class 5.3 Procedure class 5.4 Parameter classes 5.5 Worker class	<b>41</b> 42 43 44 46 47
6	6.1       Browser classes         6.2       Curves classes         6.3       Inputs classes         6.4       Listeners classes         6.5       Log classes         6.6       Manager classes         6.7       Plotter class         6.8       Qt classes         6.9       Thread classes	49 49 50 52 52 53 54 54 54

	6.11	Windows classes
7	pyme	asure.instruments 61
	7.1	Instrument classes
	7.2	Validator functions
	7.3	Comedi data acquisition
	7.4	Resource Manager
	7.5	Advantest
	7.6	Agilent
	7.7	Ametek
	7.8	AMI
	7.9	Anaheim Automation
	7.10	Anapico
	7.11	Anritsu
	7.12	Attocube
	7.13	BK Precision
	7.14	Danfysik
		Delta Elektronika
		Edwards
	7.17	Fluke
		F.W. Bell
	7.19	Hewlett Packard
	7.20	Keithley
	7.21	Keysight
	7.22	Lake Shore Cryogenics
	7.23	Newport
	7.24	National Instruments
	7.25	Oxford Instruments
	7.26	Parker
	7.27	Pendulum
	7.28	Razorbill
	7.29	Rohde & Schwarz
	7.30	Signal Recovery
	7.31	Stanford Research Systems
	7.32	Tektronix
	7.33	Thorlabs
	7.34	Yokogawa
	7.54	Tokogawa
8	Cont	ributing 219
o		8
	8.1	Using the development version
	8.2	Working on a new feature
	8.3	Making a pull-request
	8.4	Unit testing
0	Dono	nting on ones
9	керо	rting an error 223
10	Addi	ng instruments 225
		File structure
		Instrument file
		Writing properties
	10.4	Advanced properties
14	0 1	C4 JJ-
11		ng Standards 235
		Python style guides
	11.2	Documentation

	11.3 Usage of getter and setter functions	235
12	Authors	237
13	License	239
Ру	Python Module Index	
In	dex	243



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 References
- About PyMeasure

Information about development is also available:

• Getting involved

**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.

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:

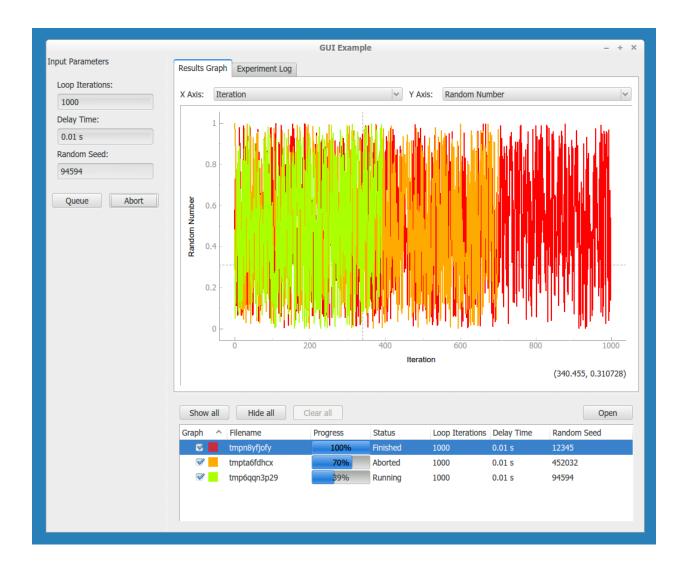
from pymeasure.instruments.keithley import Keithley2400

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.



**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 it 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 mangager 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 libaries.

## 2.2.3 Checking the version

Now that you have Python and PyMeasure installed, open up a "Jupyter Notebook" to test which version you have installed. Execute the following code into a notebook cell.

```
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 GPIB address. For this example we connect to the instrument over GPIB (using VISA) with an address of 4. See the *adapters* section below for more details.

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

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.

#### 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 class to connect to the instrument using VISA. Instead of passing a string, we 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. Unlike the VISA adapter which is specific to each instrument, the Prologix adapter can be shared by many instruments. Therefore, they are addressed separately based on the GPIB address number when passing the adapter into the instrument construction.

```
from pymeasure.adapters import PrologixAdapter

adapter = PrologixAdapter('/dev/ttyUSB0')
sourcemeter = Keithley2400(adapter.gpib(4))
```

For instruments using serial communication that have particular settings that need to be matched, a custom *Adapter* sub-class can be made. For example, the LakeShore 425 Gaussmeter connects via USB, but uses particular serial communication settings. Therefore, a *LakeShoreUSBAdapter* class enables these requirements in the background.

```
from pymeasure.instruments.lakeshore import LakeShore425
gaussmeter = LakeShore425('/dev/lakeshore425')
```

Behind the scenes the /dev/lakeshore425 port is passed to the LakeShoreUSBAdapter.

Some equipment may require the vxi-11 protocol for communication. An example would be a Agilent E5810B ethernet to GPIB bridge. To use this type equipment the python-vxi11 library has to be installed which is part of the extras package requirements.

```
from pymeasure.adapters import VXI11Adapter
from pymeasure.instruments import Instrument

adapter = VXI11Adapter("TCPIP::192.168.0.100::inst0::INSTR")
instr = Instrument(adapter, "my_instrument")
```

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) Connect to the Keithley 2400
- 4) Set up the instrument for the IV characteristic
- 5) Allocate arrays to store the resulting measurements
- 6) Loop through the current points, measure the voltage, and record
- 7) Save the final data to a CSV file
- 8) 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 = 50
max\_current = 0.01
min_current = -max_current
# Connect and configure the instrument
sourcemeter = Keithley2400("GPIB::4")
sourcemeter.reset()
sourcemeter.use_front_terminals()
sourcemeter.measure_voltage()
sourcemeter.config_current_source()
sleep(0.1) # wait here to give the instrument time to react
sourcemeter.set_buffer(averages)
# 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)
```

```
# Loop through each current point, measure and record the voltage
for i in range(data_points):
   sourcemeter.current = currents[i]
    sourcemeter.reset_buffer()
    sleep(0.1)
    sourcemeter.start_buffer()
    sourcemeter.wait_for_buffer()
    # Record the average and standard deviation
   voltages[i] = sourcemeter.means
   voltage_stds[i] = sourcemeter.standard_devs
# 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')
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):
```

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.

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):
```

```
""" 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.

#### **Modifying our script**

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
from pymeasure.experiment import IntegerParameter, FloatParameter
from time import sleep

class IVProcedure(Procedure):

   data_points = IntegerParameter('Data points', default=50)
   averages = IntegerParameter('Averages', default=50)
   max_current = FloatParameter('Maximum Current', units='A', default=0.01)
   min_current = FloatParameter('Minimum Current', units='A', default=-0.01)
```

```
DATA_COLUMNS = ['Current (A)', 'Voltage (V)', 'Voltage Std (V)']
   def startup(self):
        log.info("Connecting and configuring the instrument")
        self.sourcemeter = Keithley2400("GPIB::4")
        self.sourcemeter.reset()
        self.sourcemeter.use_front_terminals()
        self.sourcemeter.measure_voltage()
        self.sourcemeter.config_current_source()
        sleep(0.1) # wait here to give the instrument time to react
        self.sourcemeter.set_buffer(averages)
   def execute(self):
        currents = np.linspace(
            self.min_current,
            self.max current.
            num=self.data_points
        )
        # Loop through each current point, measure and record the voltage
        for current in currents:
            log.info("Setting the current to %g A" % current)
            self.sourcemeter.current = current
            self.sourcemeter.reset_buffer()
            sleep(0.1)
            self.sourcemeter.start_buffer()
            log.info("Waiting for the buffer to fill with measurements")
            self.sourcemeter.wait_for_buffer()
            self.emit('results', {
                'Current (A)': current,
                'Voltage (V)': self.sourcemeter.means,
                'Voltage Std (V)': self.sourcemeter.standard_devs
            })
            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 = 100
   procedure averages = 50
   procedure.max_current = -0.01
   procedure.min_current = 0.01
```

```
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")
```

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-flegged 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.

**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
```

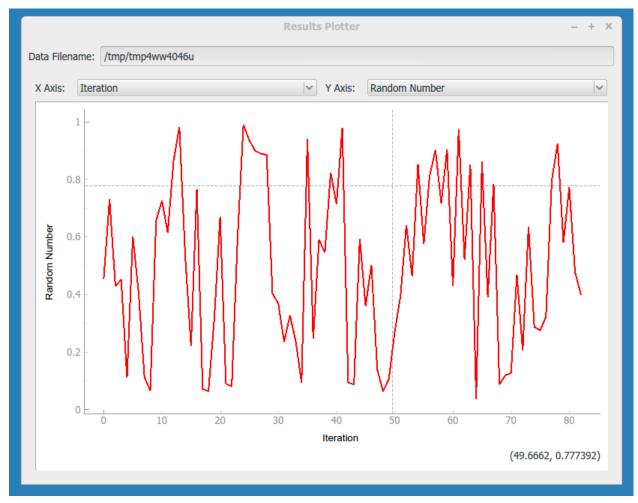
```
from pymeasure.experiment import IntegerParameter, FloatParameter, Parameter
class RandomProcedure(Procedure):
   iterations = IntegerParameter('Loop Iterations')
   delay = FloatParameter('Delay Time', units='s', default=0.2)
    seed = Parameter('Random Seed', default='12345')
   DATA_COLUMNS = ['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)
            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

18

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

Chapter 3. Tutorials

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
from pymeasure.log import console_log
from pymeasure.display.Qt import QtGui
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')
   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)
            sleep(self.delay)
            if self.should_stop():
                log.warning("Caught the stop flag in the procedure")
                break
class MainWindow(ManagedWindow):
   def __init__(self):
        super(MainWindow, self).__init__(
            procedure_class=RandomProcedure,
            inputs=['iterations', 'delay', 'seed'],
            displays=['iterations', 'delay', 'seed'],
            x_axis='Iteration',
            y_axis='Random Number'
```

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

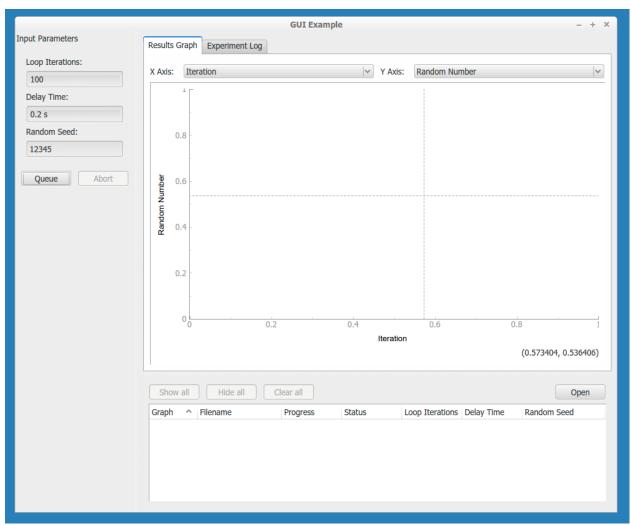
def queue(self):
    filename = tempfile.mktemp()

    procedure = self.make_procedure()
    results = Results(procedure, filename)
    experiment = self.new_experiment(results)

    self.manager.queue(experiment)

if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

This results in the following graphical display.



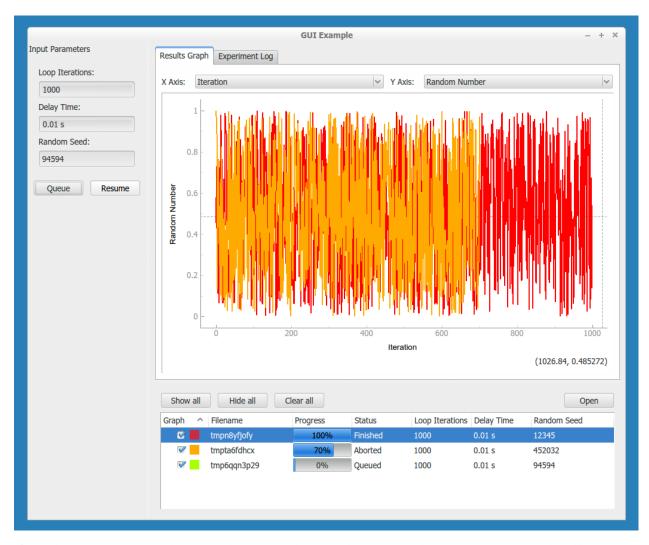
20 Chapter 3. Tutorials

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.

The queue method establishes how the Procedure object is constructed. We use the self.make\_procedure method 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.

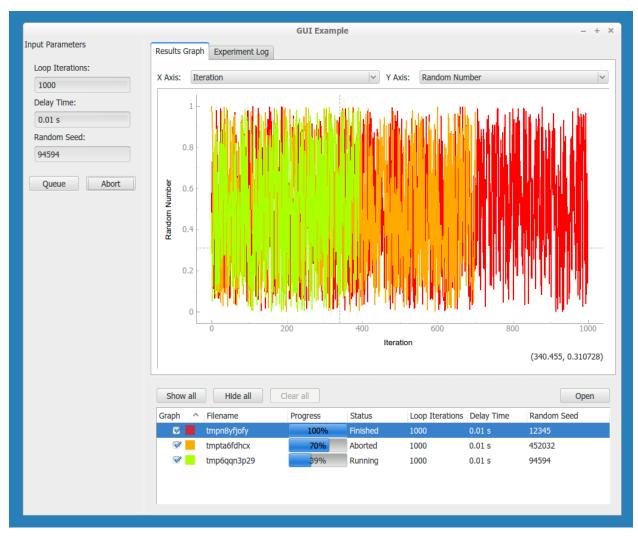


By default the Manager starts a measurement when its procedure is queued. The abort button can be pressed to stop an experiment. In the Procedure, the self.should\_stop call will catch the abort event and halt the measurement. It is important to check this value, or the Procedure will not be responsive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

22 Chapter 3. Tutorials



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.

**Note:** 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 prefered 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)
```

## 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 *setup\_plot()* 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 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

## 3.3.5 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.

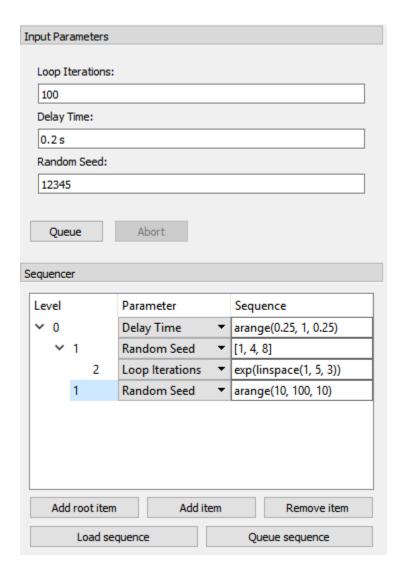
To activate the sequencer, two additional keyword arguments are added to <code>ManagedWindow</code>, namely sequencer and sequencer\_inputs. sequencer accepts a boolean stating whether or not the sequencer has to be included into the window and sequencer\_inputs accepts either <code>None</code> or a 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 MainWindow has to be modified slightly (where modified lines are marked):

```
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',
            sequencer=True,
                                                                 # Added line
                                                               # Added line
            sequencer_inputs=['iterations', 'delay', 'seed'],
            sequence_file="gui_sequencer_example_sequence.txt", # Added line, optional
        )
        self.setWindowTitle('GUI Example')
    def queue(self, procedure=None):
                                                                  # Modified line
        filename = tempfile.mktemp()
        if procedure is None:
                                                                  # Added line
            procedure = self.make_procedure()
                                                                  # Indented
        results = Results(procedure, filename)
        experiment = self.new_experiment(results)
        self.manager.queue(experiment)
```

This adds the sequencer underneath the 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 later two either add a 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 write a simple text file to quickly load a pre-defined sequence with the Load sequence button, such that the user does not need to write the sequence again each time. In the sequence file each line adds one item

26 Chapter 3. Tutorials

to the sequence tree, starting with a number of dashes (-) to indicate the 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(MainWindow, self).\_\_init\_\_ call, as was done in the example.

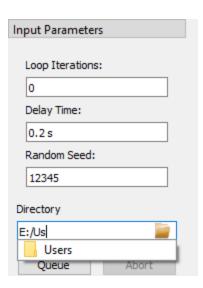
#### 3.3.6 Using the directory input

It is possible to add a directory input in order to choose where the experiment's result will be saved. This option is activated by passing a boolean key-word argument directory\_input during the <code>ManagedWindow</code> init. The value of the directory can be retrieved using the property directory.

Only the MainWindow needs to be modified in order to use this option (modified lines are marked).

```
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',
                                                                  # Added line
            directory_input=True,
        )
        self.setWindowTitle('GUI Example')
    def queue(self):
        directory = self.directory
                                                                   # Modified line
        filename = unique_filename(directory)
        results = Results(procedure, filename)
        experiment = self.new_experiment(results)
        self.manager.queue(experiment)
```

This adds the input line above the Queue and Abort buttons.



A completer is implemented allowing to quickly select an existing folder, and a button on the right side of the input widget opens a browse dialog.

#### 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.

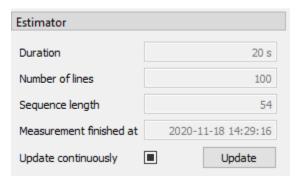
For maximum flexibility (e.g. for showing multiple and other types of estimates, such as the duration, filesize, finishing-time, etc.) it is also possible that the get\_estimates returns a list of tuples. Each of these tuple consists of two strings: the first is the name (label) of the estimate, the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

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

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

def get_estimates(self, sequence_length=None, sequence=None):
```

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 sequence (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).

30 Chapter 3. Tutorials

**CHAPTER** 

**FOUR** 

# **PYMEASURE.ADAPTERS**

The adapter classes allow the instruments to be independent of the communication method used.

Adapters for specific instruments should be grouped in an adapters.py file in the corresponding manufacturer's folder of pymeasure.instruments. For example, the adapter for communicating with LakeShore instruments over USB, <code>LakeShoreUSBAdapter</code>, is found in pymeasure.instruments.lakeshore.adapters.

# 4.1 Adapter base class

class pymeasure.adapters.Adapter(preprocess\_reply=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**

- **preprocess\_reply** optional callable used to preprocess strings received from the instrument. The callable returns the processed string.
- **kwargs** all other keyword arguments are ignored.

ask(command)

Writes the command to the instrument and returns the resulting ASCII response

Parameters command – SCPI command string to be sent to the instrument

**Returns** String ASCII response of the instrument

**binary\_values** (*command*, *header\_bytes=0*, *dtype=<class 'numpy.float32'>*)

Returns a numpy array from a query for binary data

## **Parameters**

- **command** SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header
- dtype The NumPy data type to format the values with

Returns NumPy array of values

read()

Reads until the buffer is empty and returns the resulting ASCII respone

**Returns** String ASCII response of the instrument.

values(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

write(command)

Writes a command to the instrument

**Parameters command** – SCPI command string to be sent to the instrument

# 4.2 Fake adapter

```
class pymeasure.adapters.FakeAdapter(preprocess_reply=None, **kwargs)
```

Bases: pymeasure.adapters.adapter.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() == ""
assert a.ask("10") == "10"
assert a.values("10") == [10]
```

ask(command)

Writes the command to the instrument and returns the resulting ASCII response

**Parameters command** – SCPI command string to be sent to the instrument

**Returns** String ASCII response of the instrument

**binary\_values**(*command*, *header\_bytes=0*, *dtype=<class 'numpy.float32'>*)
Returns a numpy array from a query for binary data

#### **Parameters**

- command SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header
- dtype The NumPy data type to format the values with

**Returns** NumPy array of values

read()

Returns the last commands given after the last read call.

```
values(command, separator=', ', cast=<class 'float'>, preprocess_reply=None)
```

Writes a command to the instrument and returns a list of formatted values from the result

**Parameters** 

Chapter 4. pymeasure.adapters

- **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

write(command)

Writes the command to a buffer, so that it can be read back.

# 4.3 Serial adapter

class pymeasure.adapters.SerialAdapter(port, preprocess\_reply=None, \*\*kwargs)
Bases: pymeasure.adapters.adapter.Adapter

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

#### **Parameters**

- port Serial port
- **preprocess\_reply** optional callable used to preprocess strings received from the instrument. The callable returns the processed string.
- **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 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

ask(command)

Writes the command to the instrument and returns the resulting ASCII response

**Parameters command** – SCPI command string to be sent to the instrument

**Returns** String ASCII response of the instrument

**binary\_values**(*command*, *header\_bytes*=0, *dtype*=<*class 'numpy.float32'*>)
Returns a numpy array from a query for binary data

# **Parameters**

- command SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header
- dtype The NumPy data type to format the values with

4.3. Serial adapter

**Returns** NumPy array of values

read()

Reads until the buffer is empty and returns the resulting ASCII response

**Returns** String ASCII response of the instrument.

**values**(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

write(command)

Writes a command to the instrument

**Parameters** command – SCPI command string to be sent to the instrument

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

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

#### **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

# 4.4 Prologix adapter

Bases: pymeasure.adapters.serial.SerialAdapter

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

Each PrologixAdapter is constructed based on a serial port or connection and the GPIB address to be communicated to. Serial connection sharing is achieved by using the *gpib()* method to spawn new PrologixAdapters for different GPIB addresses.

## **Parameters**

- port The Serial port name or a serial.Serial object
- address Integer GPIB address of the desired instrument
- rw\_delay An optional delay to set between a write and read call for slow to respond instruments.

- **preprocess\_reply** optional callable used to preprocess strings received from the instrument. The callable returns the processed string.
- kwargs Key-word arguments if constructing a new serial object

Variables address – Integer GPIB address of the desired instrument

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
```

**\_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 writen 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

**ask**(command)

Ask the Prologix controller, include a forced delay for some instruments.

Parameters command – SCPI command string to be sent to instrument

**binary\_values**(*command*, *header\_bytes*=0, *dtype*=<*class 'numpy.float32'*>)
Returns a numpy array from a query for binary data

#### **Parameters**

- **command** SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header
- **dtype** The NumPy data type to format the values with

**Returns** NumPy array of values

```
gpib(address, rw_delay=None)
```

Returns and PrologixAdapter 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
- rw\_delay Set a custom Read/Write delay for the instrument

**Returns** PrologixAdapter for specific GPIB address

#### read()

Reads the response of the instrument until timeout

**Returns** String ASCII response of the instrument

#### set\_defaults()

Sets up the default behavior of the Prologix-GPIB adapter

values(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

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

#### write(command)

Writes the command to the GPIB address stored in the address

**Parameters command** – SCPI command string to be sent to the instrument

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

# 4.5 VISA adapter

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

#### **Parameters**

- resource VISA resource name that identifies the address
- **visa\_library** VisaLibrary Instance, path of the VISA library or VisaLibrary spec string (@py or @ni). if not given, the default for the platform will be used.

- preprocess\_reply optional callable used to preprocess strings received from the instrument. The callable returns the processed string.
- **kwargs** Any valid key-word arguments for constructing a PyVISA instrument

#### ask(command)

Writes the command to the instrument and returns the resulting ASCII response

**Parameters command** – SCPI command string to be sent to the instrument

**Returns** String ASCII response of the instrument

## ask\_values(command, \*\*kwargs)

Writes a command to the instrument and returns a list of formatted values from the result. This leverages the *query\_ascii\_values* method in PyVISA.

#### **Parameters**

- **command** SCPI command to be sent to the instrument
- kwargs Key-word arguments to pass onto query\_ascii\_values

**Returns** Formatted response of the instrument.

**binary\_values**(*command*, *header\_bytes*=0, *dtype*=<*class 'numpy.float32'*>)
Returns a numpy array from a query for binary data

#### **Parameters**

- command SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header
- dtype The NumPy data type to format the values with

Returns NumPy array of values

# static has\_supported\_version()

Returns True if the PyVISA version is greater than 1.8

#### read()

Reads until the buffer is empty and returns the resulting ASCII response

**Returns** String ASCII response of the instrument.

# read\_bytes(size)

Reads specified number of bytes from the buffer and returns the resulting ASCII response

Parameters size – Number of bytes to read from the buffer

**Returns** String ASCII response of the instrument.

values(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

4.5. VISA adapter 37

#### 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

#### write(command)

Writes a command to the instrument

**Parameters** command – SCPI command string to be sent to the instrument

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

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

#### **Parameters**

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

Returns number of bytes written

# 4.6 VXI-11 adapter

 $\textbf{class} \ \ \textbf{pymeasure.adapters.} \textbf{VXI11Adapter} (\textit{host}, \textit{preprocess\_reply} = \textit{None}, **kwargs) \\$ 

Bases: pymeasure.adapters.adapter.Adapter

**VXII1 Adapter class. Provides a adapter object that** wraps around the read, write and ask functionality of the vxi11 library.

### **Parameters**

- **host** string containing the visa connection information.
- **preprocess\_reply** optional callable used to preprocess strings received from the instrument. The callable returns the processed string.

### ask(command)

Wrapper function for the ask command using the vx11 interface.

**Parameters** command – string with the command that will be transmitted to the instrument.

:returns string containing a response from the device.

#### ask\_raw(command)

Wrapper function for the ask\_raw command using the vx11 interface.

**Parameters command** – binary string with the command that will be transmitted to the instrument returns binary string containing the response from the device.

**binary\_values**(command, header\_bytes=0, dtype=<class 'numpy.float32'>)

Returns a numpy array from a query for binary data

#### **Parameters**

- command SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header

• **dtype** – The NumPy data type to format the values with

**Returns** NumPy array of values

# read()

Wrapper function for the read command using the vx11 interface.

return string containing a response from the device.

#### read\_raw()

Wrapper function for the read raw command using the vx11 interface.

:returns binary string containing the response from the device.

values(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

#### write(command)

Wrapper function for the write command using the vxi11 interface.

Parameters command – string with command the that will be transmitted to the instrument.

## write\_raw(command)

Wrapper function for the write\_raw command using the vxi11 interface.

Parameters command – binary string with the command that will be transmitted to the instrument

# 4.7 Telnet adapter

Bases: pymeasure.adapters.adapter.Adapter

Adapter class for using the Python telnetlib package to allow communication to instruments

### **Parameters**

- host host address of the instrument
- port TCPIP port
- query\_delay delay in seconds between write and read in the ask method
- **preprocess\_reply** optional callable used to preprocess strings received from the instrument. The callable returns the processed string.
- **kwargs** Valid keyword arguments for telnetlib. Telnet, currently this is only 'timeout'

#### ask(command)

Writes a command to the instrument and returns the resulting ASCII response

4.7. Telnet adapter 39

**Parameters command** – command string to be sent to the instrument

**Returns** String ASCII response of the instrument

**binary\_values**(*command*, *header\_bytes*=0, *dtype*=<*class 'numpy.float32'*>)
Returns a numpy array from a query for binary data

#### **Parameters**

- command SCPI command to be sent to the instrument
- header\_bytes Integer number of bytes to ignore in header
- dtype The NumPy data type to format the values with

**Returns** NumPy array of values

### read()

Read something even with blocking the I/O. After something is received check again to obtain a full reply.

**Returns** String ASCII response of the instrument.

values(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

#### write(command)

Writes a command to the instrument

Parameters command – command string to be sent to the instrument

# **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.

```
pcolor(xname, yname, zname, *args, **kwargs)
```

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

```
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\_pcolor(ax, xname, yname, zname)

Update a pcolor 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.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.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: pymeasure.thread.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: pymeasure.log.QueueListener

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

Bases: pymeasure.log.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

#### 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.

## 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

#### refresh\_parameters()

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

5.3. Procedure class 43

```
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

 $\begin{tabular}{ll} \textbf{class} & pymeasure.experiment.parameters. \textbf{BooleanParameter}(name, default=None, ui\_class=None, group\_by=None, group\_condition=True) \end{tabular}$ 

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

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

*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

**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.

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

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

Variables value – The value of the parameter

#### **Parameters**

- 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.

5.4. Parameter classes 45

• **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.

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

**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

# 5.5 Worker class

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, record)

Emits data of some topic over TCP

handle\_abort()

handle\_error()

**join**(timeout=0)

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

Parameters timeout - Timeout duration in seconds

#### 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)

# 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

## 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

5.6. Results class 47

pymeasure.experiment.results.replace\_placeholders(string, procedure, date\_format='%Y-%m-%d', time format='%H:%M:%S')

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

**CHAPTER** 

SIX

# **PYMEASURE.DISPLAY**

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

# 6.1 Browser classes

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.

Represent a row in the Browser tree widget

# 6.2 Curves classes

Creates a curve based on a predefined buffer size and allows data to be added dynamically, in additon to supporting error bars

```
append(x, y, xError=None, yError=None)
    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(*args: Any, **kwargs: Any)
    Bases: pyqtgraph.Qt.QtCore.
```

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(*args: Any, **kwargs: Any)
    Bases: pyqtgraph.
```

Creates a curve loaded dynamically from a file through the Results object and supports error bars. 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(*args: Any, **kwargs: Any)
    Bases: pyqtgraph.
```

bases. pyqegrapii.

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

```
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

# 6.3 Inputs classes

```
class pymeasure.display.inputs.BooleanInput(*args: Any, **kwargs: Any)
    Bases: pymeasure.display.inputs.Input, pyqtgraph.Qt.QtGui.
```

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.FloatInput(*args: Any, **kwargs: Any)
    Bases: pymeasure.display.inputs.Input, pyqtqraph.Qt.QtGui.
```

Spin input box for floating-point values, connected to a FloatParameter.

See also:

**Class ScientificInput** For inputs in scientific notation.

```
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.

```
class pymeasure.display.inputs.IntegerInput(*args: Any, **kwargs: Any)
```

Bases: pymeasure.display.inputs.Input, pyqtgraph.Qt.QtGui.

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.

**Parameters** parameter – parameter to connect.

```
class pymeasure.display.inputs.ListInput(*args: Any, **kwargs: Any)
```

Bases: pymeasure.display.inputs.Input, pyqtgraph.Qt.QtGui.

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(*args: Any, **kwargs: Any)
```

Bases: pymeasure.display.inputs.Input, pyqtgraph.Qt.QtGui.

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.

```
class pymeasure.display.inputs.StringInput(*args: Any, **kwargs: Any)
```

Bases: pymeasure.display.inputs.Input, pyqtgraph.Qt.QtGui.

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.

6.3. Inputs classes 51

# 6.4 Listeners classes

```
class pymeasure.display.listeners.Monitor(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtCore.
```

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

```
class pymeasure.display.listeners.QListener(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtCore.
```

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

# 6.5 Log classes

```
class pymeasure.display.log.LogHandler
     Bases: logging.Handler
     class Emitter(*args: Any, **kwargs: Any)
         Bases: pyqtgraph.Qt.QtCore.
         record
     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.

# 6.6 Manager classes

```
class pymeasure.display.manager.Experiment(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtCore.
```

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.
- browser\_item BrowserItem object

```
class pymeasure.display.manager.ExperimentQueue(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtCore.
```

Represents a Queue of Experiments and allows queries to be easily preformed

```
has_next()
```

Returns True if another item is on the queue

Returns the next experiment on the queue

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.

#### 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.

#### queue(experiment)

Adds an experiment to the queue.

## remove(experiment)

Removes an Experiment

#### resume()

Resume processing of the queue.

# 6.7 Plotter class

```
class pymeasure.display.plotter.Plotter(results, refresh_time=0.1, linewidth=1)
```

Bases: pymeasure.thread.StoppableThread

Plotter dynamically plots data from a file through the Results object and supports error bars.

## See also:

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.

6.7. Plotter class 53

# 6.8 Qt classes

All Qt imports should reference pymeasure.display.Qt, for consistant 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 accessability.

# 6.9 Thread classes

```
class pymeasure.display.thread.StoppableQThread(*args: Any, **kwargs: Any)
    Bases: pyqtgraph.Qt.QtCore.
```

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

# 6.10 Widget classes

```
class pymeasure.display.widgets.DirectoryLineEdit(*args: Any, **kwargs: Any)
```

Bases: pyqtgraph.Qt.QtGui.

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

```
class pymeasure.display.widgets.EstimatorThread(*args: Any, **kwargs: Any)
    Bases: pyqtgraph.Qt.QtCore.
```

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.ImageFrame(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtGui.
     Extends PlotFrame to plot also axis Z using colors
     ResultsClass
          alias of pymeasure.display.curves.ResultsImage
class pymeasure.display.widgets.ImageWidget(*args: Any, **kwargs: Any)
     Bases: pymeasure.display.widgets.TabWidget, pyqtgraph.Qt.QtGui.
     Extends the ImageFrame to allow different columns of the data to be dynamically choosen
     load(curve)
          Add curve to widget
     new_curve(results, color=pyqtgraph.intColor, **kwargs)
          Creates a new image
     remove(curve)
          Remove curve from widget
class pymeasure.display.widgets.InputsWidget(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtGui.
     get_procedure()
          Returns the current procedure
class pymeasure.display.widgets.LogWidget(*args: Any, **kwargs: Any)
     Bases: pymeasure.display.widgets.TabWidget, pyqtgraph.Qt.QtGui.
     Widget to display logging information in GUI
     It is recommended to include this widget in all subclasses of ManagedWindowBase
class pymeasure.display.widgets.PlotFrame(*args: Any, **kwargs: Any)
     Bases: pyqtgraph.Qt.QtGui.
     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 pymeasure.display.curves.ResultsCurve
     parse_axis(axis)
          Returns the units of an axis by searching the string
class pymeasure.display.widgets.PlotWidget(*args: Any, **kwargs: Any)
     Bases: pymeasure.display.widgets.TabWidget, pyqtgraph.Qt.QtGui.
     Extends the PlotFrame to allow different columns of the data to be dynamically choosen
     load(curve)
          Add curve to widget
     new_curve(results, color=pyqtgraph.intColor, **kwargs)
          Create a new curve
```

```
remove(curve)
```

Remove curve from widget

#### set\_color(curve, color)

Change the color of the pen of the curve

# class pymeasure.display.widgets.ResultsDialog(\*args: Any, \*\*kwargs: Any)

Bases: pyqtgraph.Qt.QtGui.

#### exception pymeasure.display.widgets.SequenceEvaluationException

Bases: Exception

Raised when the evaluation of a sequence string goes wrong.

# class pymeasure.display.widgets.SequencerWidget(\*args: Any, \*\*kwargs: Any)

Bases: pyqtgraph.Qt.QtGui.

Widget that allows to generate a sequence of measurements with varying parameters. Moreover, one can write a simple text file to easily load a sequence.

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

```
static eval_string(string, name=None, depth=None)
```

Evaluate the given string. The string is evaluated using a list of pre-defined functions that are deemed safe to use, to prevent the execution of malicious code. For this purpose, also any built-in functions or global variables are not available.

#### **Parameters**

- **string** String to be interpreted.
- name Name of the to-be-interpreted string, only used for error messages.
- **depth** Depth of the to-be-interpreted string, only used for error messages.

## get\_sequence\_from\_tree()

Generate a list of parameters from the sequence tree.

# 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.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 the specialized implementation of these method's

#### load(curve)

Add curve to widget

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

Create a new curve

### remove(curve)

Remove curve from widget

### set\_color(curve, color)

Set color for widget

# 6.11 Windows classes

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

class pymeasure.display.windows.ManagedWindow(\*args: Any, \*\*kwargs: Any)
 Bases: pyqtgraph.Qt.QtGui.

Display experiment output with an PlotWidget class.

#### See also:

**Tutorial** *Using the ManagedWindow* A tutorial and example on the basic configuration and usage of Managed-Window.

## **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
- linewidth linewidth for the displayed curves, default is 1
- \*\*kwargs optional keyword arguments that will be passed to ManagedWindowBase

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, etc.

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

- LogWidget
- PlotWidget
- ImageWidget

6.11. Windows classes 57

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 Managed-Window.

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
- **directory\_input** specify, if present, where the experiment's result will be saved.
- **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)

Abstract method, which must be overridden by the child class.

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:

```
def queue(self):
    filename = unique_filename('results', prefix="data") # from pymeasure.
    →experiment

    procedure = self.make_procedure() # Procedure class was passed at_
    →construction
    results = Results(procedure, filename)
    experiment = self.new_experiment(results)

self.manager.queue(experiment)
```

# 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.

```
class pymeasure.display.windows.PlotterWindow(*args: Any, **kwargs: Any)
    Bases: pyqtgraph.Qt.QtGui.
```

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

6.11. Windows classes 59

# **PYMEASURE.INSTRUMENTS**

This section contains documentation on the instrument classes.

# 7.1 Instrument classes

class pymeasure.instruments.Instrument(adapter, name, includeSCPI=True, \*\*kwargs)

This provides the base class for all Instruments, which is independent of the particular Adapter used to connect for communication to the instrument. It provides basic SCPI commands by default, but can be toggled with includeSCPI.

#### **Parameters**

- adapter An Adapter object
- **name** A string name
- includeSCPI A boolean, which toggles the inclusion of standard SCPI commands

## ask(command)

Writes the command to the instrument through the adapter and returns the read response.

**Parameters command** – command string to be sent to the instrument

## check errors()

Read all errors from the instrument.

**Returns** list of error entries

#### clear()

Clears the instrument status byte

#### property complete

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.

Returns a property for the class based on the supplied commands. This property may be set and read from the instrument.

#### **Parameters**

-  $\ensuremath{\text{get\_command}}$  – A string command that asks for the value

- **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
- get\_process A function that take a value and allows processing before value mapping, returning the processed value
- **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
- check\_get\_errors Toggles checking errors after getting

#### property id

Requests and returns the identification of the instrument.

Returns 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 take a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that take a command and allows processing before executing the command, for both getting and setting
- **check\_get\_errors** Toggles checking errors after getting

#### property options

Requests and returns the device options installed.

# read()

Reads from the instrument through the adapter and returns the response.

#### reset()

Resets the instrument.

Returns 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

## shutdown()

Brings the instrument to a safe and stable state

#### property status

Requests and returns the status byte and Master Summary Status bit.

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

Reads a set of values from the instrument through the adapter, passing on any key-word arguments.

#### write(command)

Writes the command to the instrument through the adapter.

**Parameters command** – command string to be sent to the instrument

```
class pymeasure.instruments.Mock(wait=0.1, **kwargs)
```

Bases: pymeasure.instruments.instrument.Instrument

Mock instrument for testing.

## get\_time()

Get elapsed time

# get\_voltage()

Get the voltage.

## get\_wave()

Get wave.

#### reset\_time()

Reset the timer to 0 s.

## set\_output\_voltage(value)

Set the voltage.

# set\_time(value)

Wait for the timer to reach the specified time. If value = 0, reset.

### property time

Get elapsed time

# property voltage

Get the voltage.

## property wave

Get wave.

7.1. Instrument classes 63

# 7.2 Validator functions

Validators are used in conjunction with the *Instrument.control* function to allow properties with complex restrictions for valid values. They are described in more detail in the *Advanced properties* 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)

Join a list of validators together as a single. Expects a list of validator functions and values.

#### **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

**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 the maximum and greater than the minimum of the range. 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

# 7.3 Comedi data acquisition

The Comedi libraries provide a convenient method for interacting with data acquisition cards, but are restricted to Linux compatible operating systems.

```
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

# 7.4 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:

# 7.5 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*.

# 7.5.1 Advantest R3767CG Vector Network Analyzer

Bases: pymeasure.instruments.instrument.Instrument

Represents the Advantest R3767CG VNA. Implements controls to change the analysis range and to retreve the data for the trace.

```
property center_frequency
Center Frequency in Hz

property id
Reads the instrument identification

property span_frequency
Span Frequency in Hz
```

property start\_frequency

Starting frequency in Hz

property stop\_frequency
Stoping frequency in Hz

1 6 1 7

property trace\_1

Reads the Data array from trace 1 after formatting

# 7.6 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*.

# 7.6.1 Agilent 8257D Signal Generator

```
class pymeasure.instruments.agilent.Agilent8257D(adapter, **kwargs)
    Bases: pymeasure.instruments.instrument.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 precent, 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 precentage
- **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 inernal 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.

# 7.6.2 Agilent 8722ES Vector Network Analyzer

# class pymeasure.instruments.agilent.Agilent8722ES(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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.

# 7.6.3 Agilent E4408B Spectrum Analyzer

## class pymeasure.instruments.agilent.AgilentE4408B(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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).

# 7.6.4 Agilent E4980 LCR Meter

# class pymeasure.instruments.agilent.AgilentE4980(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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: Resitance [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

# 7.6.5 Agilent 34410A Multimeter

# class pymeasure.instruments.agilent.Agilent34410A(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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

AC current, in Amps

### property current\_dc

DC current, in Amps

### property resistance

Resistance, in Ohms

# property resistance\_4w

Four-wires (remote sensing) resistance, in Ohms

### property voltage\_ac

AC voltage, in Volts

# property voltage\_dc

DC voltage, in Volts

# 7.6.6 HP/Agilent/Keysight 34450A Digital Multimeter

# class pymeasure.instruments.agilent.Agilent34450A(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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

Reads a capacitance measurement in Farads, based on the active mode.

### property capacitance\_auto\_range

A boolean property that toggles auto ranging for capacitance.

# property capacitance\_range

A property that controls 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, 100E-9, 1E-6, 10E-6, 100E-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, 10E3, 1E6, 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

Reads a continuity measurement in Ohms, based on the active mode.

### property current

Reads a DC current measurement in Amps, based on the active mode.

# property current\_ac

Reads an AC current measurement in Amps, based on the active mode.

### property current\_ac\_auto\_range

A boolean property that toggles auto ranging for AC current.

#### property current\_ac\_range

A property that controls 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

An property that controls 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

A boolean property that toggles auto ranging for DC current.

#### property current\_range

A property that controls 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

A property that controls 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

Reads a diode measurement in Volts, based on the active mode.

### property frequency

Reads a frequency measurement in Hz, based on the active mode.

### property frequency\_aperture

A property that controls 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

A boolean property that toggles auto ranging for AC current in frequency measurements.

### property frequency\_current\_range

A property that controls 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

A boolean property that toggles auto ranging for AC voltage in frequency measurements.

### property frequency\_voltage\_range

A property that controls 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 resistance

Reads a resistance measurement in Ohms for 2-wire configuration, based on the active mode.

### property resistance\_4w

Reads a resistance measurement in Ohms for 4-wire configuration, based on the active mode.

### property resistance\_4w\_auto\_range

A boolean property that toggles auto ranging for 4-wire resistance.

### property resistance\_4w\_range

A property that controls the 4-wire resistance range in Ohms, which can take values 100, 1E3, 10E3, 10E3, 10E6, 10E6, 10E6, 10E6, as well as "MIN", "MAX", or "DEF" (1E3). Auto-range is disabled when this property is set.

# property resistance\_4w\_resolution

A property that controls 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

A boolean property that toggles auto ranging for 2-wire resistance.

### property resistance\_range

A property that controls 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

A property that controls 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

Reads a temperature measurement in Celsius, based on the active mode.

#### property voltage

Reads a DC voltage measurement in Volts, based on the active mode.

## property voltage\_ac

Reads an AC voltage measurement in Volts, based on the active mode.

# property voltage\_ac\_auto\_range

A boolean property that toggles auto ranging for AC voltage.

# property voltage\_ac\_range

A property that controls 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

A property that controls 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

A boolean property that toggles auto ranging for DC voltage.

### property voltage\_range

A property that controls 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

A property that controls 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).

# 7.6.7 Agilent 4155/4156 Semiconductor Parameter Analyzer

Represents the Agilent 4155/4156 Semiconductor Parameter Analyzer and provides a high-level interface for taking current-voltage (I-V) measurements.

(continues on next page)

(continued from previous page)

```
# take measurements
status = smu.measure()

# measured data is a pandas dataframe and can be exported to csv.
data = smu.get_data(path='./t1.csv')
```

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 *configure()* 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
```

(continues on next page)

(continued from previous page)

```
},

"VAR2": {
    "start" : 0,
    "step" : 10e-6,
    "points" : 3,
    "compliance" : 2
}
```

# property analyzer\_mode

A string property that controls the instrument operating mode.

• Values: SWEEP, SAMPLING

```
smu.analyzer_mode = "SWEEP"
```

# configure(config\_file)

Convenience function to configure the channel setup and sweep using a JSON (JavaScript Object Notation) configuration file.

**Parameters config\_file** – JSON file to configure instrument channels.

```
instr.configure('config.json')
```

### property data\_variables

Gets a string list of data variables for which measured data is available. This looks for all the variables saved by the *save()* and *save\_var()* methods and returns it. This is useful for creation of dataframe headers.

**Returns** List

```
header = instr.data_variables
```

## property delay\_time

A floating point property that 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()

Disables all channels in the instrument.

```
instr.disable_all()
```

# get\_data(path=None)

Gets 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

A floating point property that 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

A string property that controls the integration time.

• Values: SHORT, MEDIUM, LONG

```
instr.integration_time = "MEDIUM"
```

# measure(period='INF', points=100)

Performs a single measurement and waits 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 <code>save()</code>.

**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()

Stops the ongoing measurement

```
instr.stop()
```

class pymeasure.instruments.agilent.agilent4156.SMU(resourceName, channel, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

# property channel\_function

A string property that controls the SMU<n> channel function.

• Values: VAR1, VAR2, VARD or CONS.

```
instr.smu1.channel_function = "VAR1"
```

#### property channel\_mode

A string property that controls the SMU<n> channel mode.

• Values: V, I or COMM

VPULSE AND IPULSE are not yet supported.

```
instr.smu1.channel_mode = "V"
```

# property compliance

This command sets 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

This command sets 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

Define 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.current_name = "Ibase"
```

## property disable

This command deletes the settings of SMU<n>.

```
instr.smu1.disable()
```

### property series\_resistance

This command controls the series resistance of SMU<n>.

• Values: 00HM, 10KOHM, 100KOHM, or 1MOHM

```
instr.smu1.series_resistance = "10KOHM"
```

# property voltage\_name

Define 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.VAR1(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.agilent.agilent4156.VARX

Class to handle all the specific definitions needed for VAR1. Most common methods are inherited from base class.

#### property spacing

This command selects the sweep type of VAR1.

• Values: LINEAR, LOG10, LOG25, LOG50.

class pymeasure.instruments.agilent.agilent4156.VAR2(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.agilent.agilent4156.VARX

Class to handle all the specific definitions needed for VAR2. Common methods are imported from base class.

# property points

This command sets 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(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Class to handle all the definitions needed for VARD. VARD is always defined in relation to VAR1.

# property compliance

This command sets the sweep COMPLIANCE value of VARD.

```
instr.vard.compliance = 0.1
```

# property offset

This command sets 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

This command sets 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(resourceName, var\_name, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Base class to define sweep variable settings

# property compliance

Sets the sweep COMPLIANCE value.

```
instr.var1.compliance = 0.1
```

# property start

Sets the sweep START value.

```
instr.var1.start = 0
```

# property step

Sets the sweep STEP value.

```
instr.var1.step = 0.1
```

### property stop

Sets the sweep STOP value.

```
instr.var1.stop = 3
```

class pymeasure.instruments.agilent.agilent4156.VMU(resourceName, channel, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

# property channel\_mode

A string property that controls the VMU<n> channel mode.

• Values: V, DVOL

# property disable

This command disables the settings of VMU<n>.

```
instr.vmu1.disable()
```

# property voltage\_name

Define the voltage name of the VMU 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.vmu1.voltage_name = "Vanode"
```

class pymeasure.instruments.agilent.agilent4156.VSU(resourceName, channel, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

# property channel\_function

A string property that controls the VSU channel function.

• Value: VAR1, VAR2, VARD or CONS.

## property channel\_mode

Get channel mode of VSU<n>.

# property constant\_value

This command sets the constant source value of VSU<n>.

```
instr.vsu1.constant_value = 0
```

# property disable

This command deletes the settings of VSU<n>.

```
instr.vsu1.disable()
```

### property voltage\_name

Define the voltage name of the VSU 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.vsu1.voltage_name = "Ve"
```

# 7.6.8 Agilent 33220A Arbitrary Waveform Generator

class pymeasure.instruments.agilent.Agilent33220A(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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
                                # Enable burst mode
wfg.burst_state = True
                                # A burst will consist of 10 cycles
wfg.burst_ncycles = 10
wfg.burst_mode = "TRIGGERED"
wfg.trigger_source = "BUS"
                                # A burst will be applied on a trigger
                                # A burst will be triggered on TRG*
wfg.output = True
                                # Enable output of waveform generator
wfg.trigger()
                                # Trigger a burst
wfg.wait_for_trigger()
                                 # Wait until the triggering is finished
wfg.beep()
                                 # "beep"
print(wfg.check_errors())
                                 # Get the error queue
```

### property amplitude

A floating point property that controls the voltage amplitude of the output waveform in V, from 10e-3 V to 10 V. Can be set.

### property amplitude\_unit

A string property that controls the units of the amplitude. Valid values are Vpp (default), Vrms, and dBm. Can be set.

# beep()

Causes a system beep.

# property beeper\_state

A boolean property that controls the state of the beeper. Can be set.

### property burst\_mode

A string property that controls the burst mode. Valid values are: TRIG<GERED>, GAT<ED>. This setting can be set.

# property burst\_ncycles

An integer property that sets the number of cycles to be output when a burst is triggered. Valid values are 1 to 50000. This can be set.

### property burst\_state

A boolean property that controls whether the burst mode is on (True) or off (False). Can be set.

### property frequency

A floating point property that controls the frequency of the output waveform in Hz, from 1e-6 (1 uHz) to 20e+6 (20 MHz), depending on the specified function. Can be set.

### property offset

A floating point property that controls the voltage offset of the output waveform in V, from 0 V to 4.995 V,

depending on the set voltage amplitude (maximum offset = (10 - voltage) / 2). Can be set.

#### property output

A boolean property that turns on (True) or off (False) the output of the function generator. Can be set.

### property pulse\_dutycycle

A floating point property that controls the duty cycle of a pulse waveform function in percent. Can be set.

### property pulse\_hold

A string property that controls if either the pulse width or the duty cycle is retained when changing the period or frequency of the waveform. Can be set to: WIDT<H> or DCYC<LE>.

### property pulse\_period

A floating point property that controls the period of a pulse waveform function in seconds, ranging from 200 ns to 2000 s. Can be set and 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

A floating point property that controls the 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. Can be set.

### property pulse\_width

A floating point property that controls the width of a pulse waveform function in seconds, ranging from 20 ns to 2000 s, within a set of restrictions depending on the period. Can be set.

#### property ramp\_symmetry

A floating point property that controls the symmetry percentage for the ramp waveform. Can be set.

### property remote\_local\_state

A string property that controls the remote/local state of the function generator. Valid values are: LOC<AL>, REM<OTE>, RWL<OCK>. This setting can only be set.

# property shape

A string property that controls the output waveform. Can be set to: SIN<USOID>, SQU<ARE>, RAMP, PULS<E>, NOIS<E>, DC, USER.

### property square\_dutycycle

A floating point property that controls the duty cycle of a square waveform function in percent. Can be set.

# trigger()

Send a trigger signal to the function generator.

# property trigger\_source

A string property that controls the trigger source. Valid values are: IMM<EDIATE> (internal), EXT<ERNAL> (rear input), BUS (via trigger command). This setting can be set.

#### property trigger\_state

A boolean property that controls whether the output is triggered (True) or not (False). Can be set.

# property voltage\_high

A floating point property that controls the upper voltage of the output waveform in V, from -4.990 V to 5 V (must be higher than low voltage). Can be set.

### property voltage\_low

A floating point property that controls the lower voltage of the output waveform in V, from -5 V to 4.990 V (must be lower than high voltage). Can be set.

# 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.

# 7.6.9 Agilent 33500 Function/Arbitrary Waveform Generator Family

```
class pymeasure.instruments.agilent.Agilent33500(adapter, **kwargs)
    Bases: pymeasure.instruments.instrument.Instrument
```

Represents the Agilent 33500 Function/Arbitrary Waveform Generator family. Individual devices are represented by subclasses.

```
generator = Agilent33500("GPIB::1")
generator.shape = 'SIN'
                                        # Sets the output signal shape to sine
generator.frequency = 1e3
                                        # Sets the output frequency to 1 kHz
                                        # Sets the output amplitude to 5 Vpp
generator.amplitude = 5
generator.output = 'on'
                                        # Enables the output
                                        # Set shape to arbitrary
generator.shape = 'ARB'
generator.arb_srate = 1e6
                                        # Set sample rate to 1MSa/s
generator.data_volatile_clear()
                                        # Clear volatile internal memory
generator.data_arb(
                                        # Send data points of arbitrary waveform
    'test',
    range(-10000, 10000, +20),
                                        # In this case a simple ramp
                                        # Data format is set to 'DAC'
    data_format='DAC'
generator.arb_file = 'test'
                                        # Select the transmitted waveform 'test'
```

### property amplitude

A floating point property that controls the voltage amplitude of the output waveform in V, from 10e-3 V to 10 V. Depends on the output impedance. Can be set.

### property amplitude\_unit

A string property that controls the units of the amplitude. Valid values are VPP (default), VRMS, and DBM. Can be set.

### property arb\_advance

A string property that selects how the device advances from data point to data point. Can be set to 'TRIG<GER>' or 'SRAT<E>' (default).

### property arb\_file

A string property that selects the arbitrary signal from the volatile memory of the device. String has to match an existing arb signal in volatile memore (set by data\_arb()). Can be set.

# property arb\_filter

A string property that selects the filter setting for arbitrary signals. Can be set to 'NORM<AL>', 'STEP' and 'OFF'.

### 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 (maximum range, can be lower depending on your device). This can be set.

#### beep()

Causes a system beep.

# property burst\_mode

A string property that controls the burst mode. Valid values are: TRIG<GERED>, GAT<ED>. This setting can be set.

### property burst\_ncycles

An integer property that sets the number of cycles to be output when a burst is triggered. Valid values are 1 to 100000. This can be set.

### property burst\_period

A floating point property that controls the period of subsequent bursts. Has to follow the equation burst\_period > (burst\_ncycles / frequency) + 1  $\mu$ s. Valid values are 1  $\mu$ s to 8000 s. Can be set.

### property burst\_state

A boolean property that controls whether the burst mode is on (True) or off (False). Can be set.

# 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). TODO: *Binary is not yet implemented* 

#### **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 the 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

A string property which is displayed on the front panel of the device. Can be set.

# property ext\_trig\_out

A boolean property that controls whether the trigger out signal is active (True) or not (False). This signal is output from the Ext Trig connector on the rear panel in Burst and Wobbel mode. Can be set.

### property frequency

A floating point property that controls the frequency of the output waveform in Hz, from 1 uHz to 120 MHz (maximum range, can be lower depending on your device), depending on the specified function. Can be set.

#### property id

Reads the instrument identification

# property offset

A floating point property that controls the voltage offset of the output waveform in V, from 0 V to 4.995 V, depending on the set voltage amplitude (maximum offset = (Vmax - voltage) / 2). Can be set.

#### property output

A boolean property that turns on (True, 'on') or off (False, 'off') the output of the function generator. Can be set.

### property output\_load

Sets the expected load resistance (should be the load impedance connected to the output. The output impedance is always 50 Ohm, this setting can be used to correct the displayed voltage for loads unmatched to 50 Ohm. Valid values are between 1 and 10 kOhm or INF for high impedance. No validator is used since both numeric and string inputs are accepted, thus a value outside the range will not return an error. Can be set.

# property phase

A floating point property that controls the phase of the output waveform in degrees, from -360 degrees to 360 degrees. Not available for arbitrary waveforms or noise. Can be set.

### property pulse\_dutycycle

A floating point property that controls the duty cycle of a pulse waveform function in percent, from 0% to 100%. Can be set.

#### property pulse hold

A string property that controls if either the pulse width or the duty cycle is retained when changing the period or frequency of the waveform. Can be set to: WIDT<H> or DCYC<LE>.

### property pulse\_period

A floating point property that controls the period of a pulse waveform function in seconds, ranging from 33 ns to 1e6 s. Can be set and 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

A floating point property that controls the edge time in seconds for both the rising and falling edges. It is defined as the time between the 10% and 90% thresholds of the edge. Valid values are between 8.4 ns to 1  $\mu s$ . Can be set.

# property pulse\_width

A floating point property that controls the width of a pulse waveform function in seconds, ranging from 16 ns to 1e6 s, within a set of restrictions depending on the period. Can be set.

### property ramp\_symmetry

A floating point property that controls the symmetry percentage for the ramp waveform, from 0.0% to 100.0% Can be set.

# property shape

A string property that controls the output waveform. Can be set to: SIN<USOID>, SQU<ARE>, TRI<ANGLE>, RAMP, PULS<E>, PRBS, NOIS<E>, ARB, DC.

### property square\_dutycycle

A floating point property that controls the duty cycle of a square waveform function in percent, from 0.01% to 99.98%. The duty cycle is limited by the frequency and the minimal pulse width of 16 ns. See manual for more details. Can be set.

# trigger()

Send a trigger signal to the function generator.

### property trigger\_source

A string property that controls the trigger source. Valid values are: IMM<EDIATE> (internal), EXT<ERNAL> (rear input), BUS (via trigger command). This setting can be set.

# property voltage\_high

A floating point property that controls the upper voltage of the output waveform in V, from -4.990 V to 5 V (must be higher than low voltage by at least 1 mV). Can be set.

### property voltage\_low

A floating point property that controls the lower voltage of the output waveform in V, from -5 V to 4.990 V (must be lower than high voltage by at least 1 mV). Can be set.

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.

# 7.6.10 Agilent 33521A Function/Arbitrary Waveform Generator

class pymeasure.instruments.agilent.Agilent33521A(adapter, \*\*kwargs)
 Bases: pymeasure.instruments.agilent.agilent33500.Agilent33500

Represents the Agilent 33521A Function/Arbitrary Waveform Generator. This documentation page shows only methods different from the parent class *Agilent33500*.

# 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.

# 7.6.11 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
BC	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	force_gnd(), SMU.force_gnd()
ERRX?	check_errors()
FL	SMU.filter
FMT	data_format()
*IDN?	id()
*LRN?	<pre>query_learn(), multiple methods to read/format settings directly</pre>
MI	SMU.sampling_source() mode: 'CURRENT'
ML	sampling_mode
MM	meas_mode()
MSC	<pre>sampling_auto_abort()</pre>
MT	<pre>sampling_timing()</pre>
MV	SMU.sampling_source() mode: 'VOLTAGE'
*0PC?	check_idle()
PA	pause()
PAD	parallel_meas
RI	meas_range_current
RM	SMU.meas_range_current_auto()
*RST	reset()
RV	meas_range_voltage
	continues on next nage

continues on next page

Command	Property/Method
SSR	series_resistor
TSC	time_stamp
TSR	<pre>clear_timer()</pre>
UNT?	<pre>query_modules()</pre>
WAT	<pre>wait_time()</pre>
WI	<pre>SMU.staircase_sweep_source() mode: 'CURRENT'</pre>
WM	<pre>sweep_auto_abort()</pre>
WSI	SMU.synchronous_sweep_source() mode: 'CURRENT'
WSV	SMU.synchronous_sweep_source() mode: 'VOLTAGE'
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',
... timeout=600000)
# query SMU config from instrument and initialize all SMU instances
b1500.initialize_all_smus()
# set data output format (required!)
b1500.data_format(21, mode=1) #call after SMUs are initialized to get names for the ...
... channels
```

### IV measurement with 4 SMUs

(continues on next page)

(continued from previous page)

```
#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
# 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 beacuse 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

```
# choose measurement mode
b1500.meas_mode('SAMPLING', *b1500.smu_references) #order in smu_references determines_
→order of measurement
number_of_channels = len(b1500.smu_references)
# settings for individual SMUs
for smu in b1500.smu_references:
    smu.enable() #enable SMU
    smu.adc_type = 'HSADC' #set ADC to high-speed ADC
    smu.meas_range_current = '1 nA'
    smu.meas_op_mode = 'COMPLIANCE_SIDE' # other choices: Current, Voltage, FORCE_SIDE,_
→ 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)
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.

```
class pymeasure.instruments.agilent.agilentB1500.AgilentB1500(resourceName, **kwargs)
```

Bases: pymeasure.instruments.instrument.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 Excecution 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 (\*OPC?)

### 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)

Type 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 returned)

# 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 intrument.

### 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. Halfs 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

## 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 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.

```
\textbf{class} \texttt{ pymeasure.instruments.agilent.agilentB1500.CustomIntEnum} (\textit{value})
```

Bases: enum.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

```
class pymeasure.instruments.agilent.agilentB1500.ADCType(value)
```

Bases: pymeasure.instruments.agilent.agilentB1500.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)

 $Bases: \ pymeasure.instruments.agilent.agilentB1500.CustomIntEnum$ 

ADC Mode

```
AUTO = 0
    MANUAL = 1
    PLC = 2
    TIME = 3
class pymeasure.instruments.agilent.agilentB1500.AutoManual(value)
    Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
    Auto/Manual selection
    AUTO = 0
    MANUAL = 1
class pymeasure.instruments.agilent.agilentB1500.MeasMode(value)
    Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
    Measurement Mode
    SPOT = 1
    STAIRCASE\_SWEEP = 2
    SAMPLING = 10
class pymeasure.instruments.agilent.agilentB1500.MeasOpMode(value)
    Bases: pymeasure.instruments.agilent.agilentB1500.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)
    Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
    Sweep Mode
    LINEAR_SINGLE = 1
    LOG_SINGLE = 2
    LINEAR_DOUBLE = 3
    LOG_DOUBLE = 4
class pymeasure.instruments.agilent.agilentB1500.SamplingMode(value)
    Bases: pymeasure.instruments.agilent.agilentB1500.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)
     Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
     Output after sampling
     BASE = 1
     BIAS = 2
class pymeasure.instruments.agilent.agilentB1500.StaircaseSweepPostOutput(value)
     Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
     Output after staircase sweep
     START = 1
     STOP = 2
class pymeasure.instruments.agilent.agilentB1500.CompliancePolarity(value)
     Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
     Compliance polarity
     AUTO = 0
     MANUAL = 1
\textbf{class} \ \ \textbf{pymeasure.instruments.agilent.agilentB1500.WaitTimeType} (\textit{value})
     Bases: pymeasure.instruments.agilent.agilentB1500.CustomIntEnum
     Wait time type
     SMU_SOURCE = 1
     SMU\_MEASUREMENT = 2
     CMU\_MEASUREMENT = 3
```

## 7.7 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*.

7.7. Ametek 103

## 7.7.1 Ametek 7270 DSP Lockin Amplifier

## class pymeasure.instruments.ametek.Ametek7270(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

This is the class for the Ametek DSP 7270 lockin amplifier

### property adc1

Reads the input value of ADC1 in Volts

### property adc2

Reads the input value of ADC2 in Volts

## property adc3

Reads the input value of ADC3 in Volts

## property adc4

Reads the input value of ADC4 in Volts

## property dac1

A floating point property that represents the output value on DAC1 in Volts. This property can be set.

## property dac2

A floating point property that represents the output value on DAC2 in Volts. This property can be set.

## property dac3

A floating point property that represents the output value on DAC3 in Volts. This property can be set.

## property dac4

A floating point property that represents the output value on DAC4 in Volts. This property can be set.

### property frequency

A floating point property that represents the lock-in frequency in Hz. This property can be set.

### property harmonic

An integer property that represents the reference harmonic mode control, taking values from 1 to 127. This property can be set.

### property id

Reads the instrument identification

## property mag

Reads the magnitude in Volts

## property phase

A floating point property that represents the reference harmonic phase in degrees. This property can be set.

## property sensitivity

A floating point property that controls the sensitivity range in Volts, which can take discrete values from 2 nV to 1 V. This property can be set.

## set\_channel\_A\_mode()

Sets instrument to channel A mode – assuming it is in voltage mode

### set\_differential\_mode(lineFiltering=True)

Sets instrument to differential mode – assuming it is in voltage mode

## set\_voltage\_mode()

Sets instrument to voltage control mode

#### shutdown()

Ensures the instrument in a safe state

### property slope

A integer property that controls the filter slope in dB/octave, which can take the values 6, 12, 18, or 24 dB/octave. This property can be set.

## property time\_constant

A floating point property that controls the time constant in seconds, which takes values from 10 microseconds to 100,000 seconds. This property can be set.

### property voltage

A floating point property that represents the voltage in Volts. This property can be set.

### property x

Reads the X value in Volts

### property x1

Reads the first harmonic X value in Volts

### property x2

Reads the second harmonic X value in Volts

### property xy

Reads both the X and Y values in Volts

### property y

Reads the Y value in Volts

## property y1

Reads the first harmonic Y value in Volts

#### property y2

Reads the second harmonic Y value in Volts

## 7.8 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*.

## 7.8.1 AMI 430 Power Supply

```
class pymeasure.instruments.ami.AMI430(resourceName, **kwargs)
    Bases: pymeasure.instruments.instrument.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
```

(continues on next page)

7.8. AMI 105

(continued from previous page)

```
magnet.pause # Pauses the ramping
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

A floating point property that sets the coil contant in kGauss/A.

## disable\_persistent\_switch()

Disables the persistent switch.

### enable\_persistent\_switch()

Enables the persistent switch.

## property field

Reads the field in kGauss of the magnet.

## has\_persistent\_switch\_enabled()

Returns a boolean if the persistent switch is enabled.

### property magnet\_current

Reads the current in Amps of the magnet.

### 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

A floating point property that sets the current ramping rate in A/s.

### property ramp\_rate\_field

A floating point property that sets the field ramping rate in kGauss/s.

## 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

Reads the field in kGauss of the magnet.

## property supply\_current

Reads the current in Amps of the power supply.

## property target\_current

A floating point property that sets the target current in A for the magnet.

## property target\_field

A floating point property that sets the target field in kGauss for the magnet.

## property voltage\_limit

A floating point property that sets the voltage limit for charging/discharging the magnet.

```
wait_for_holding(should_stop=<function AM1430.<lambda>>, timeout=800, interval=0.1)
zero()
```

Initiates the ramping of the magnetic field to zero current/field with ramping rate previously set.

## 7.9 Anaheim Automation

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*.

## 7.9.1 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.

class pymeasure.instruments.anaheimautomation.DPSeriesMotorController(resourceName,

address=0, encoder\_enabled=False, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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.

### ask(command)

Override the instrument base ask method to add the motor controller's address to the command string.

**Parameters command** – command string to be sent to the instrument

## 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 the 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 encoder\_enabled 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 *absolute position* 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.

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

Override the instrument base values method to add the motor controller's address to the command string.

**Parameters command** – command string to be sent to 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.

## 7.10 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*.

## 7.10.1 Anapico APSIN12G Signal Generator

class pymeasure.instruments.anapico.APSIN12G(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Anapico APSIN12G Signal Generator with option 9K, HP and GPIB.

#### property blanking

A string property that represents the blanking of output power when frequency is changed. ON makes the output to be blanked (off) while changing frequency. This property can be set.

### disable\_rf()

Disables the RF output.

#### enable\_rf()

Enables the RF output.

## property frequency

A floating point property that represents the output frequency in Hz. This property can be set.

## property power

A floating point property that represents the output power in dBm. This property can be set.

### property reference\_output

A string property that represents the 10MHz reference output from the synth. This property can be set.

7.10. Anapico 109

## 7.11 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*.

## 7.11.1 Anritsu MG3692C Signal Generator

## class pymeasure.instruments.anritsu.AnritsuMG3692C(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Anritsu MG3692C Signal Generator

### disable()

Disables the signal output.

### enable()

Enables the signal output.

### property frequency

A floating point property that represents the output frequency in Hz. This property can be set.

### property output

A boolean property that represents the signal output state. This property can be set to control the output.

### property power

A floating point property that represents the output power in dBm. This property can be set.

### shutdown()

Shuts down the instrument, putting it in a safe state.

## 7.11.2 Anritsu MS9710C Optical Spectrum Analyzer

## class pymeasure.instruments.anritsu.AnritsuMS9710C(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Anritsu MS9710C Optical Spectrum Analyzer.

## property analysis

Analysis Control

## property analysis\_result

Read back analysis result from current scan.

## property average\_point

Number of averages to take on each point (2-1000), or OFF

## property average\_sweep

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

Returns the data condition of data memory register A. Starting wavelength, and a sampling point (11, 12, n).

#### property data\_memory\_a\_size

Returns the number of points sampled in data memory register A.

### property data\_memory\_a\_values

Reads the binary data from memory register A.

## property data\_memory\_b\_condition

Returns the data condition of data memory register B. Starting wavelength, and a sampling point (11, 12, n).

### property data\_memory\_b\_size

Returns the number of points sampled in data memory register B.

#### property data\_memory\_b\_values

Reads the binary data from memory register B.

## property data\_memory\_select

Memory Data Select.

## property dip\_search

Dip Search Mode

### property ese2

Extended Event Status Enable Register 2

## property esr2

Extended Event Status Register 2

## property level\_lin

Level Linear Scale (/div)

### property level\_log

Level Log Scale (/div)

## property level\_opt\_attn

Optical Attenuation Status (ON/OFF)

## property level\_scale

Current Level Scale

### property measure\_mode

Returns the current Measure Mode the OSA is in.

## measure\_peak()

Measure the peak and return the trace marker.

## property peak\_search

Peak Search Mode

## read\_memory(slot='A')

Read the scan saved in a memory slot.

### property resolution

Resolution (nm)

## property resolution\_actual

Resolution Actual (ON/OFF)

### property resolution\_vbw

Video Bandwidth Resolution

### property sampling\_points

Number of sampling points

## single\_sweep(\*\*kwargs)

Perform a single sweep and wait for completion.

7.11. Anritsu 111

### property trace\_marker

Sets the trace marker with a wavelength. Returns the trace wavelength and power.

### property trace\_marker\_center

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

Center Wavelength of Spectrum Scan in nm.

## property wavelength\_marker\_value

Wavelength Marker Value (wavelength or freq.?)

### property wavelength\_span

Wavelength Span of Spectrum Scan in nm.

### property wavelength\_start

Wavelength Start of Spectrum Scan in nm.

## property wavelength\_stop

Wavelength Stop of Spectrum Scan in nm.

### property wavelength\_value\_in

Wavelength value in Vacuum or Air

### property wavelengths

Return a numpy array of the current wavelengths of scans.

## 7.12 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*.

## 7.12.1 Attocube Adapters

Bases: pymeasure.adapters.telnet.TelnetAdapter

Adapter class for connecting to the Attocube Standard Console. This console is a Telnet prompt with password authentication.

### **Parameters**

- host host address of the instrument
- **port** TCPIP port
- passwd password required to open the connection
- kwargs Any valid key-word argument for TelnetAdapter

### ask(command)

Writes a command to the instrument and returns the resulting ASCII response

**Parameters command** – command string to be sent to the instrument

**Returns** String ASCII response of the instrument

### check\_acknowledgement(reply, msg=")

checks the last reply of the instrument to be 'OK', otherwise a ValueError is raised.

#### **Parameters**

- reply last reply string of the instrument
- msg optional message for the eventual error

## extract\_value(reply)

preprocess\_reply function for the Attocube console. This function tries to extract <value> from 'name = <value> [unit]'. If <value> can not be identified the original string is returned.

Parameters reply - reply string

**Returns** string with only the numerical value, or the original string

### read()

Reads a reply of the instrument which consists of two or more lines. The first ones are the reply to the command while the last one is 'OK' or 'ERROR' to indicate any problem. In case the reply is not OK a ValueError is raised.

**Returns** String ASCII response of the instrument.

write(command, check ack=True)

Writes a command to the instrument

### **Parameters**

- **command** command string to be sent to the instrument
- **check\_ack** boolean flag to decide if the acknowledgement is read back from the instrument. This should be True for set pure commands and False otherwise.

## 7.12.2 Attocube ANC300 Motion Controller

Bases: pymeasure.instruments.instrument.Instrument

Attocube ANC300 Piezo stage controller with several axes

### **Parameters**

- host host address of the instrument
- 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 delay between sending and reading (default 0.05 sec)
- kwargs Any valid key-word argument for TelnetAdapter

## property controllerBoardVersion

Serial number of the controller board

7.12. Attocube 113

### ground\_all()

Grounds all axis of the controller.

### stop\_all()

Stop all movements of the axis.

## property version

Version number and instrument identification

### class pymeasure.instruments.attocube.anc300.Axis(controller, axis)

Bases: object

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

Saved capacity value in nF of the axis.

### property frequency

Frequency of the stepping motion in Hertz from 1 to 10000 Hz. This property can be set.

### 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

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 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

## property offset\_voltage

Offset voltage in Volts from 0 to 150 V. This property can be set.

### property output\_voltage

Output voltage in volts.

## property pattern\_down

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

step up pattern of the piezo drive. 256 values ranging from 0 to 255 representing the 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

Serial number of the axis

## property stepd

Step downwards for N steps. Mode must be 'stp' and N must be positive.

#### property stepu

Step upwards for N steps. Mode must be 'stp' and N must be positive.

#### stop()

Stop any motion of the axis

## property voltage

Amplitude of the stepping voltage in volts from 0 to 150 V. This property can be set.

## 7.13 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*.

## 7.13.1 BK Precision 9130B DC Power Supply

## class pymeasure.instruments.bkprecision.BKPrecision9130B(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the BK Precision 9130B DC Power Supply interface for interacting with the instrument.

### property channel

An integer property used to control which channel is selected. Can only take values [1, 2, 3].

### check\_errors()

Read all errors from the instrument.

Returns list of error entries

## property complete

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

Floating point property used to control current of the selected channel.

#### property id

Requests and returns the identification of the instrument.

## property options

Requests and returns the device options installed.

#### reset()

Resets the instrument.

## shutdown()

Brings the instrument to a safe and stable state

### property source\_enabled

A boolean property that controls whether the source is enabled, takes values True or False.

7.13. BK Precision 115

#### property status

Requests and returns the status byte and Master Summary Status bit.

### property voltage

Floating point property used to control voltage of the selected channel.

## 7.14 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*.

## 7.14.1 Danfysik Serial Adapter

## class pymeasure.instruments.danfysik.DanfysikAdapter(port)

Bases: pymeasure.adapters.serial.SerialAdapter

Provides a SerialAdapter with the specific baudrate and timeout for Danfysik serial communication.

Initiates the adapter to open serial communcation over the supplied port.

**Parameters port** – A string representing the serial port

#### read()

Overwrites the SerialAdapter.read method to automatically 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

### write(command)

Overwrites the SerialAdapter.write method to automatically append a Unix-style linebreak at the end of the command.

**Parameters command** – SCPI command string to be sent to the instrument

## 7.14.2 Danfysik 8500 Power Supply

## class pymeasure.instruments.danfysik.Danfysik8500(port)

Bases: pymeasure.instruments.instrument.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

The actual current in Amps. This property can be set through *current\_ppm*.

### property current\_ppm

The current in parts per million. This property can be set.

## property current\_setpoint

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

Reads 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

The polarity of the current supply, being either -1 or 1. This property can be set by suppling one of these values.

## ramp\_to\_current(current, points, delay\_time=1)

Executes set\_ramp\_to\_current() and starts the ramp.

#### 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.

7.14. Danfysik 117

### **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

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

A list of human-readable strings that contain the instrument status information, based on status\_hex.

### property status\_hex

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

## 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

## 7.15 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*.

## 7.15.1 Delta Elektronica SM7045D Power source

 $\textbf{class} \ \ \textbf{pymeasure.instruments.deltaelektronika.SM7045D} (\textit{resourceName}, **kwargs)$ 

Bases: pymeasure.instruments.instrument.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)

7.15. Delta Elektronika 119

### 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.

## 7.16 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*.

## 7.16.1 Edwards nxds vacuum pump

pymeasure.instruments.edwards.nxds

## **7.17 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*.

## 7.17.1 Fluke 7341 Temperature bath

class pymeasure.instruments.fluke.Fluke7341(resource name, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the compact constant temperature bath from Fluke

## property id

Read the instrument model

### property set\_point

A *float* property to set the bath temperature set-point. Valid values are in the range –40 to 150 °C. The unit is as defined in property *unit*. This property can be read

## property temperature

Read the current bath temperature. The unit is as defined in property unit.

### property unit

A string property that controls the temperature unit. Possible values are c for Celsius and f for Fahrenheit`.

## 7.18 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*.

## 7.18.1 F.W. Bell 5080 Handheld Gaussmeter

## class pymeasure.instruments.fwbell.FWBell5080(port)

Bases: pymeasure.instruments.instrument.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

### ask(command)

Overwrites the *Instrument.ask* method to remove the last 2 characters from the output.

### auto\_range()

Enables the auto range functionality.

### property field

Reads a floating point value of the field in the appropriate units.

```
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

Reads the idenfitication information.

## property range

A floating point property that controls the maximum field range in the active units. This can take the values of 300 G, 3 kG, and 30 kG for Gauss, 30 mT, 300 mT, and 3 T for Tesla, and 23.88 kAm, 238.8 kAm, and 2388 kAm for Amp-meter.

### read()

Overwrites the *Instrument.read* method to remove the last 2 characters from the output.

### reset()

Resets the instrument.

## property units

A string property that controls the field units, 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.

7.18. F.W. Bell 121

#### values(command)

Overwrites the Instrument.values method to remove the lastv2 characters from the output.

## 7.19 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*.

## 7.19.1 HP 33120A Arbitrary Waveform Generator

## class pymeasure.instruments.hp.HP33120A(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Hewlett Packard 33120A Arbitrary Waveform Generator and provides a high-level interface for interacting with the instrument.

### property amplitude

A floating point property that controls the voltage amplitude of the output signal. The default units are in peak-to-peak Volts, but can be controlled by *amplitude\_units*. The allowed range depends on the waveform shape and can be queried with *max\_amplitude* and *min\_amplitude*.

## property amplitude\_units

A string property that controls the units of the amplitude, which can take the values Vpp, Vrms, dBm, and default.

### beep()

Causes a system beep.

## property frequency

A floating point property that controls 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.

### property max\_amplitude

Reads the maximum amplitude in Volts for the given shape

## property max\_frequency

Reads the maximum frequency in Hz for the given shape

### property max\_offset

Reads the maximum *offset* in Volts for the given shape

### property min\_amplitude

Reads the minimum amplitude in Volts for the given shape

## property min\_frequency

Reads the minimum frequency in Hz for the given shape

#### property min\_offset

Reads the minimum offset in Volts for the given shape

## property offset

A floating point property that controls 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

A string property that controls the shape of the wave, which can take the values: sinusoid, square, triangle, ramp, noise, dc, and user.

## 7.19.2 HP 34401A Multimeter

## class pymeasure.instruments.hp.HP34401A(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the HP 34401A instrument.

### property current\_ac

AC current, in Amps

## property current\_dc

DC current, in Amps

## property resistance

Resistance, in Ohms

## property resistance\_4w

Four-wires (remote sensing) resistance, in Ohms

### property voltage\_ac

AC voltage, in Volts

## property voltage\_dc

DC voltage, in Volts

## 7.19.3 HP 3478A Multimeter

## class pymeasure.instruments.hp.HP3478A(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Hewlett Packard 3748A 5 1/2 digit multimeter and provides a high-level interface for interacting with the instrument.

### class ERRORS(value)

Bases: enum.IntFlag

Enum element for errror bit decoding

## GPIB\_trigger()

Initate trigger via low-level GPIB-command (aka GET - group execute trigger)

## class SRQ(value)

Bases: enum.IntFlag

Enum element for SRQ mask 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

7.19. Hewlett Packard 123

### 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\_enabled

Return calibration enable switch setting, based on front-panel selector switch

	Value	Status
	True	calbration possible
Ì	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

## classmethod decode\_mode(function)

Method to decode current mode

**Parameters function** – int indicating the measurement function selected

**Return cur\_mode** string with the current measurement mode

Rtype cur\_mode str

## classmethod decode\_range(range\_undecoded, function)

Method to decode current range

## **Parameters**

- range\_undecoded int to be decoded
- function int indicating the measurement function selected

**Return cur\_range** float value repesenting the active measurment range

Rtype cur\_range float

## classmethod decode\_status(status\_bytes, field=None)

Method to handle the decoding of the status bytes into something meaningfull

### **Parameters**

- status\_bytes list of bytes to be decoded
- **field** name of field to be returned

#### Return ret val int status value

## static decode\_trigger(status\_bytes)

Method to decode trigger mode

Parameters status\_bytes - list of bytes to be decoded

Return trigger mode string with the current trigger mode

Rtype trigger\_mode str

## 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

### get\_status()

Method to read the status bytes from the instrument :return current\_status: a byte array representing the instrument status :rtype current\_status: bytes

## 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 properly can be set. Allowed values are

Mode	e 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.

7.19. Hewlett Packard 125

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

## reset()

Initatiates a reset (like a power-on reset) of the HP3478A

### 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)

## shutdown()

provides a way to gracefully close the connection to the HP3478A

## property status

Returns an object representing the current status of the unit.

## 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	

## 7.20 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*.

## 7.20.1 Keithley 2000 Multimeter

class pymeasure.instruments.keithley.Keithley2000(adapter, \*\*kwargs)

 $Bases: \quad \textit{pymeasure.} instruments. instrument. Instrument, \quad \text{pymeasure.} instruments. keithley. \\ buffer. Keithley Buffer$ 

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

A string property that enables or disables the system status beeper, which can take the values: :code:'enabled' and :code:'disabled'.

### property buffer\_data

Returns a numpy array of values from the buffer.

### property buffer\_points

An integer property that controls 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

### property complete

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)

Configures 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

Reads a DC or AC current measurement in Amps, based on the active *mode*.

## property current\_ac\_bandwidth

A floating point property that sets the AC current detector bandwidth in Hz, which can take the values 3, 30, and 300 Hz.

### property current\_ac\_digits

An integer property that controls the number of digits in the AC current readings, which can take values from 4 to 7.

### property current\_ac\_nplc

A floating point property that controls 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.

7.20. Keithley 127

## property current\_ac\_range

A floating point property that controls 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

A floating point property that controls the AC current reference value in Amps, which can take values from -3.1 to 3.1 A.

## property current\_digits

An integer property that controls the number of digits in the DC current readings, which can take values from 4 to 7.

## property current\_nplc

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls the DC current reference value in Amps, which can take values from -3.1 to 3.1 A.

## disable\_buffer()

Disables 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

Reads a frequency measurement in Hz, based on the active *mode*.

### property frequency\_aperature

A floating point property that controls 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

An integer property that controls the number of digits in the frequency readings, which can take values from 4 to 7.

### property frequency\_reference

A floating point property that controls the frequency reference value in Hz, which can take values from 0 to 15 MHz.

## property frequency\_threshold

A floating point property that controls the voltage signal threshold level in Volts for the frequency measurement, which can take values from 0 to 1010 V.

### property id

Requests and returns the identification of the instrument.

## is\_buffer\_full()

Returns 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 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

### 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

A string property that controls the configuration mode for measurements, which can take the values:

7.20. Keithley 129

:code:'current' (DC), :code:'current ac', :code:'voltage' (DC), :code:'voltage ac', :code:'resistance' (2-wire), :code:'resistance 4W' (4-wire), :code:'period', :code:'frequency', :code:'temperature', :code:'diode', and :code:'frequency'.

## property options

Requests and returns the device options installed.

### property period

Reads a period measurement in seconds, based on the active *mode*.

## property period\_aperature

A floating point property that controls 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

An integer property that controls the number of digits in the period readings, which can take values from 4 to 7.

## property period\_reference

A floating point property that controls the period reference value in seconds, which can take values from 0 to 1 s.

### property period\_threshold

A floating point property that controls the voltage signal threshold level in Volts for the period measurement, which can take values from 0 to 1010 V.

### remote()

Places the instrument in the remote state, which is does not need to be explicity 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()

Resets the buffer.

### property resistance

Reads a resistance measurement in Ohms for both 2-wire and 4-wire configurations, based on the active *mode*.

## property resistance\_4W\_digits

An integer property that controls the number of digits in the 4-wire resistance readings, which can take values from 4 to 7.

### property resistance\_4W\_nplc

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls the 4-wire resistance reference value in Ohms, which can take values from 0 to 120 MOhms.

### property resistance\_digits

An integer property that controls the number of digits in the 2-wire resistance readings, which can take values from 4 to 7.

## property resistance\_nplc

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls 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

Requests and returns the status byte and Master Summary Status bit.

### stop\_buffer()

Aborts the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

### property temperature

Reads a temperature measurement in Celsius, based on the active *mode*.

### property temperature\_digits

An integer property that controls the number of digits in the temperature readings, which can take values from 4 to 7.

### property temperature\_nplc

A floating point property that controls 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

A floating point property that controls the temperature reference value in Celsius, which can take values from -200 to 1372 C.

#### property trigger\_count

An integer property that controls the trigger count, which can take values from 1 to 9,999.

## property trigger\_delay

A floating point property that controls the trigger delay in seconds, which can take values from 1 to 9,999,999.999 s.

#### property voltage

Reads a DC or AC voltage measurement in Volts, based on the active *mode*.

### property voltage\_ac\_bandwidth

A floating point property that sets the AC voltage detector bandwidth in Hz, which can take the values 3, 30, and 300 Hz.

7.20. Keithley 131

### property voltage\_ac\_digits

An integer property that controls the number of digits in the AC voltage readings, which can take values from 4 to 7.

## property voltage\_ac\_nplc

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls the AC voltage reference value in Volts, which can take values from -757.5 to 757.5 Volts.

## property voltage\_digits

An integer property that controls the number of digits in the DC voltage readings, which can take values from 4 to 7.

### property voltage\_nplc

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls the DC voltage reference value in Volts, which can take values from -1010 to 1010 V.

## wait\_for\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Blocks 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

## 7.20.2 Keithley 2260B DC Power Supply

## class pymeasure.instruments.keithley.Keithley2260B(adapter, read\_termination=\n', \*\*kwargs)

Bases: pymeasure.instruments.instrument.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.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

Simultaneous control of voltage (volts) and current (amps). 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()

Logs any system errors reported by the instrument.

## property complete

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

Reads the current (in Ampere) the dc power supply is putting out.

### property current\_limit

A floating point property that controls 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.

### property enabled

A boolean property that controls whether the source is enabled, takes values True or False.

### property error

Returns a tuple of an error code and message from a single error.

## property id

Requests and returns the identification of the instrument.

#### property options

Requests and returns the device options installed.

## property power

Reads the power (in Watt) the dc power supply is putting out.

### reset()

Resets the instrument.

#### shutdown()

Disable output, call parent function

## property status

Requests and returns the status byte and Master Summary Status bit.

## property voltage

Reads the voltage (in Volt) the dc power supply is putting out.

### property voltage\_setpoint

A floating point property that controls 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.

7.20. Keithley 133

## 7.20.3 Keithley 2400 SourceMeter

class pymeasure.instruments.keithley.Keithley2400(adapter, \*\*kwargs)

 $Bases: \quad \textit{pymeasure.} instruments. instrument. Instrument, \quad \text{pymeasure.} instruments. keithley. \\ buffer. Keithley Buffer$ 

Represents the Keithely 2400 SourceMeter and provides 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
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)

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

### property auto\_output\_off

A boolean property that enables or disables the auto output-off. Valid values are True (output off after measurement) and False (output stays on after measurement).

## auto\_range\_source()

Configures the source to use an automatic range.

### property auto\_zero

A property that controls the auto zero option. Valid values are True (enabled) and False (disabled) and 'ONCE' (force immediate).

## 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

Returns a numpy array of values from the buffer.

## property buffer\_points

An integer property that controls the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

#### check\_errors()

Logs any system errors reported by the instrument.

## property complete

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

A floating point property that controls the compliance current in Amps.

### property compliance\_voltage

A floating point property that controls the compliance voltage in Volts.

### config\_buffer(points=64, delay=0)

Configures 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

Reads the current in Amps, if configured for this reading.

## property current\_nplc

A floating point property that controls 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

A floating point property that controls 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()

Disables 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

A boolean property that controls 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

Returns a tuple of an error code and message from a single error.

7.20. Keithley 135

## property filter\_count

A integer property that controls the number of readings that are acquired and stored in the filter buffer for the averaging

## property filter\_state

A string property that controls if the filter is active.

#### property filter\_type

A String property that controls the filter's type. REP: Repeating filter MOV: Moving filter

## property id

Requests and returns the identification of the instrument.

### is\_buffer\_full()

Returns True if the buffer is full of measurements.

### property line\_frequency

An integer property that controls the line frequency in Hertz. Valid values are 50 and 60.

### property line\_frequency\_auto

A boolean property that enables or disables auto line frequency. Valid values are True and False.

#### property max\_current

Returns the maximum current from the buffer

### property max\_resistance

Returns the maximum resistance from the buffer

### property max\_voltage

Returns the maximum voltage from the buffer

### property maximums

Returns the calculated maximums for voltage, current, and resistance from the buffer data as a list.

### property mean\_current

Returns the mean current from the buffer

### property mean\_resistance

Returns the mean resistance from the buffer

## property mean\_voltage

Returns the mean voltage from the buffer

## property means

Reads the calculated means (averages) for voltage, current, and resistance from the buffer data as a list.

## property measure\_concurent\_functions

A boolean property that enables or disables the ability to measure more than one function simultaneously. When disabled, volts function is enabled. Valid values are True and False.

### 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

Returns the minimum current from the buffer

## property min\_resistance

Returns the minimum resistance from the buffer

## property min\_voltage

Returns the minimum voltage from the buffer

## property minimums

Returns the calculated minimums for voltage, current, and resistance from the buffer data as a list.

#### property options

Requests and returns the device options installed.

## property output\_off\_state

Select 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 0.5% full scale of the present current range, whichever is greater.

### 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

### 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.

7.20. Keithley 137

### **Parameters**

- target\_voltage A voltage in Amps
- **steps** An integer number of steps
- pause A pause duration in seconds to wait between steps

#### reset()

Resets the instrument and clears the queue.

## reset\_buffer()

Resets the buffer.

## property resistance

Reads the resistance in Ohms, if configured for this reading.

## property resistance\_nplc

A floating point property that controls 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

A floating point property that controls the resistance range in Ohms, which can take values from 0 to 210 MOhms. Auto-range is disabled when this property is set.

### sample\_continuously()

Causes the instrument to continuously read samples and turns off any buffer or output triggering

### 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

## set\_trigger\_counts(arm, trigger)

Sets 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()

Ensures that the current or voltage is turned to zero and disables the output.

#### property source\_current

A floating point property that controls the source current in Amps.

## property source\_current\_range

A floating point property that controls 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\_delay

A floating point property that sets 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

A boolean property that enables or disables auto delay. Valid values are True and False.

### property source\_enabled

A boolean property that controls 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

A string property that controls 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

A floating point property that controls the source voltage in Volts.

# property source\_voltage\_range

A floating point property that controls 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

Returns the calculated standard deviations for voltage, current, and resistance from the buffer data as a list.

# start\_buffer()

Starts the buffer.

#### status()

Requests and returns the status byte and Master Summary Status bit.

#### property std\_current

Returns the current standard deviation from the buffer

# property std\_resistance

Returns the resistance standard deviation from the buffer

#### property std\_voltage

Returns the voltage standard deviation from the buffer

#### stop\_buffer()

Aborts 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.

#### property trigger\_count

An integer property that controls the trigger count, which can take values from 1 to 9,999.

# property trigger\_delay

A floating point property that controls the trigger delay in seconds, which can take values from 0 to 999.9999 s.

#### 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

### 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

Reads the voltage in Volts, if configured for this reading.

# property voltage\_nplc

A floating point property that controls 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

A floating point property that controls 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\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Blocks 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

An integer property that controls the number of wires in use for resistance measurements, which can take the value of 2 or 4.

# 7.20.4 Keithley 2450 SourceMeter

```
class pymeasure.instruments.keithley.Keithley2450(adapter, **kwargs)
```

 $Bases: \quad \textit{pymeasure.} instruments. instrument. Instrument, \quad \text{pymeasure.} instruments. keithley. \\ buffer. Keithley Buffer$ 

Represents the Keithely 2450 SourceMeter and provides a high-level interface for interacting with the instrument.

```
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
\mbox{keithley.compliance\_voltage} \ = \ 10 \ \ \ \ \ \mbox{\# 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
keithley.shutdown()
                                         # Ramps the current to 0 mA and disables.
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

Returns a numpy array of values from the buffer.

# property buffer\_points

An integer property that controls the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

#### check errors()

Logs any system errors reported by the instrument.

# property complete

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

A floating point property that controls the compliance current in Amps.

# property compliance\_voltage

A floating point property that controls the compliance voltage in Volts.

# config\_buffer(points=64, delay=0)

Configures 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

Reads the current in Amps, if configured for this reading.

#### property current\_filter\_count

A integer property that controls the number of readings that are acquired and stored in the filter buffer for the averaging

# property current\_filter\_state

A string property that controls if the filter is active.

#### property current\_filter\_type

A String property that controls the filter's type for the current. REP: Repeating filter MOV: Moving filter

# property current\_nplc

A floating point property that controls 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

Select 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 0.5% full scale of the present current range, whichever is greater.

#### property current\_range

A floating point property that controls 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()

Disables 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

Returns a tuple of an error code and message from a single error.

# property id

Requests and returns the identification of the instrument.

#### is\_buffer\_full()

Returns True if the buffer is full of measurements.

#### property max\_current

Returns the maximum current from the buffer

### property max\_resistance

Returns the maximum resistance from the buffer

# property max\_voltage

Returns the maximum voltage from the buffer

#### property maximums

Returns the calculated maximums for voltage, current, and resistance from the buffer data as a list.

#### property mean\_current

Returns the mean current from the buffer

# property mean\_resistance

Returns the mean resistance from the buffer

#### property mean\_voltage

Returns the mean voltage from the buffer

# property means

Reads 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

Returns the minimum current from the buffer

# property min\_resistance

Returns the minimum resistance from the buffer

# property min\_voltage

Returns the minimum voltage from the buffer

#### property minimums

Returns the calculated minimums for voltage, current, and resistance from the buffer data as a list.

# property options

Requests and returns 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

#### reset()

Resets the instrument and clears the queue.

# reset\_buffer()

Resets the buffer.

#### property resistance

Reads the resistance in Ohms, if configured for this reading.

#### property resistance\_nplc

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls the source current in Amps.

# property source\_current\_delay

A floating point property that sets 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

A boolean property that enables or disables auto delay. Valid values are True and False.

# property source\_current\_range

A floating point property that controls 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

Reads a boolean value that is True if the source is enabled.

#### property source\_mode

A string property that controls 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

A floating point property that controls the source voltage in Volts.

#### property source\_voltage\_delay

A floating point property that sets 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

A boolean property that enables or disables auto delay. Valid values are True and False.

#### property source\_voltage\_range

A floating point property that controls 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

Returns the calculated standard deviations for voltage, current, and resistance from the buffer data as a list.

#### start\_buffer()

Starts the buffer.

# property status

Requests and returns the status byte and Master Summary Status bit.

# property std\_current

Returns the current standard deviation from the buffer

#### property std\_resistance

Returns the resistance standard deviation from the buffer

# property std\_voltage

Returns the voltage standard deviation from the buffer

#### stop\_buffer()

Aborts 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

Reads the voltage in Volts, if configured for this reading.

# property voltage\_filter\_count

A integer property that controls the number of readings that are acquired and stored in the filter buffer for the averaging

# property voltage\_filter\_type

A String property that controls the filter's type for the current. REP: Repeating filter MOV: Moving filter

# property voltage\_nplc

A floating point property that controls 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

Select 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 0.5% full scale of the present current range, whichever is greater.

# property voltage\_range

A floating point property that controls 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\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Blocks 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

An integer property that controls the number of wires in use for resistance measurements, which can take the value of 2 or 4.

# 7.20.5 Keithley 2700 MultiMeter/Switch System

# class pymeasure.instruments.keithley.Keithley2700(adapter, \*\*kwargs)

 $Bases: \quad \textit{pymeasure.} instruments. instrument. Instrument, \quad \text{pymeasure.} instruments. keithley. \\ buffer. Keithley Buffer$ 

Represents the Keithely 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

Returns a numpy array of values from the buffer.

# property buffer\_points

An integer property that controls 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()

Logs any system errors reported by the instrument.

#### 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

Parameter that controls the opened and closed channels. All mentioned channels are closed, other channels will be opened.

# property complete

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)

Configures 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()

Disables 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

A string property that controls 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

Returns a tuple of an error code and message from a single error.

# 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

Requests and returns the identification of the instrument.

#### is\_buffer\_full()

Returns True if the buffer is full of measurements.

# open\_all\_channels()

Open all channels of the Keithley 2700.

# property open\_channels

A parameter that opens 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

Property that lists the installed cards in the Keithley 2700. Returns a dict with the integer card numbers on the position.

#### reset()

Resets the instrument and clears the queue.

# reset\_buffer()

Resets the buffer.

#### shutdown()

Brings the instrument to a safe and stable state

### start\_buffer()

Starts the buffer.

# property status

Requests and returns the status byte and Master Summary Status bit.

#### stop\_buffer()

Aborts the buffering measurement, by stopping the measurement arming and triggering sequence. If possible, a Selected Device Clear (SDC) is used.

# property text\_enabled

A boolean property that controls 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\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Blocks 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

# 7.20.6 Keithley 6221 AC and DC Current Source

class pymeasure.instruments.keithley.Keithley6221(adapter, \*\*kwargs)

 $Bases: \quad \textit{pymeasure.} instruments. instrument. Instrument, \quad \text{pymeasure.} instruments. keithley. \\ buffer. Keithley Buffer$ 

Represents the Keithely 6221 AC and DC current source and provides a high-level interface for interacting with the instrument.

```
keithley = Keithley6221("GPIB::1")
keithley.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.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
keithlev.waveform arm()
                                               # Arm (load) the waveform
keithley.waveform_start()
                                               # Start the waveform
keithley.adapter.wait_for_srq()
                                               # Wait for the pulse to finish
keithley.waveform_abort()
                                               # Disarm (unload) the waveform
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

Returns a numpy array of values from the buffer.

# property buffer\_points

An integer property that controls the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

#### check\_errors()

Logs any system errors reported by the instrument.

# property complete

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)

Configures 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.

# disable\_buffer()

Disables 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

A boolean property that controls 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

Returns a tuple of an error code and message from a single error.

#### property id

Requests and returns the identification of the instrument.

# is\_buffer\_full()

Returns True if the buffer is full of measurements.

# property measurement\_event\_enabled

An integer value that controls 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

An integer value that reads 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 operation\_event\_enabled

An integer value that controls 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

An integer value that reads 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

Requests and returns the device options installed.

# property output\_low\_grounded

A boolean property that controls 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

An integer value that controls 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

An integer value that reads 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.

#### reset()

Resets the instrument and clears the queue.

#### reset\_buffer()

Resets 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

#### shutdown()

Disables the output.

#### property source\_auto\_range

A boolean property that controls the auto range of the current source. Valid values are True or False.

# property source\_compliance

A floating point property that controls the compliance of the current source in Volts. valid values are in range 0.1 [V] to 105 [V].

# property source\_current

A floating point property that controls the source current in Amps.

#### property source\_delay

A floating point property that sets 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

A boolean property that controls 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

A floating point property that controls 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

An integer value that controls 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

An integer value that controls 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

An integer value that reads 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

Requests and returns the status byte and Master Summary Status bit.

#### stop buffer()

Aborts 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\_buffer(should stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Blocks 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

A floating point property that controls 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

A floating point property that controls 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

A floating point property that controls the duration of the waveform in seconds. Valid values are in range 100e-9 to 999999.999.

# property waveform\_dutycycle

A floating point property that controls 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

A floating point property that controls the frequency of the waveform in Hertz. Valid values are in range 1e-3 to 1e5.

# property waveform\_function

A string property that controls the selected wave function. Valid values are "sine", "ramp", "square", "arbitrary1", "arbitrary2", "arbitrary3" and "arbitrary4".

# property waveform\_offset

A floating point property that controls the offset of the waveform in Amps. Valid values are in range -0.105 to 0.105.

# property waveform\_ranging

A string property that controls 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

A boolean property that controls 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.

# 7.20.7 Keithley 6517B Electrometer

#### class pymeasure.instruments.keithley.Keithley6517B(adapter, \*\*kwargs)

```
Bases: \quad \textit{pymeasure.instruments.instrument.} Instrument, \quad \texttt{pymeasure.instruments.keithley.} \\ buffer. \texttt{KeithleyBuffer}
```

Represents the Keithely 6517B ElectroMeter and provides a high-level interface for interacting with the instrument.

(continues on next page)

(continued from previous page)

```
keithley.source_voltage = 20  # Sets the source voltage to 20 V
keithley.enable_source()  # Enables the source output

keithley.measure_resistance()  # Sets up to measure resistance

keithley.ramp_to_voltage(50)  # Ramps the voltage to 50 V
print(keithley.resistance)  # Prints the resistance in Ohms

keithley.shutdown()  # Ramps the voltage to 0 V
# 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

Returns a numpy array of values from the buffer.

# property buffer\_points

An integer property that controls the number of buffer points. This does not represent actual points in the buffer, but the configuration value instead.

#### check\_errors()

Logs any system errors reported by the instrument.

# property complete

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)

Configures 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

Reads the current in Amps, if configured for this reading.

# property current\_nplc

A floating point property that controls 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

A floating point property that controls 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()

Disables 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

Returns a tuple of an error code and message from a single error.

#### property id

Requests and returns the identification of the instrument.

# is\_buffer\_full()

Returns 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 attribut

# 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 attribut

# 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 attribut

#### property options

Requests and returns 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

# reset()

Resets the instrument and clears the queue.

#### reset\_buffer()

Resets the buffer.

# property resistance

Reads the resistance in Ohms, if configured for this reading.

#### property resistance\_nplc

A floating point property that controls 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

A floating point property that controls 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

Boolean property which enables or disables resistance current limit

# property source\_enabled

Reads a boolean value that is True if the source is enabled.

# property source\_voltage

A floating point property that controls the source voltage in Volts.

#### property source\_voltage\_range

A floating point property that controls 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

Requests and returns the status byte and Master Summary Status bit.

#### stop\_buffer()

Aborts 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

Reads the voltage in Volts, if configured for this reading.

# property voltage\_nplc

A floating point property that controls 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

A floating point property that controls 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\_buffer(should\_stop=<function KeithleyBuffer.<lambda>>, timeout=60, interval=0.1)

Blocks 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

# 7.20.8 Keithley 2750 Multimeter/Switch System

# class pymeasure.instruments.keithley.Keithley2750(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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

# close(channel)

Closes (connects) the specified channel.

**Parameters channel** (int) – 3-digit number for the channel

Returns None

# property closed\_channels

Reads the list of closed channels

# property complete

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

Requests and returns the identification of the instrument.

# 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

Requests and returns the device options installed.

### reset()

Resets the instrument.

# shutdown()

Brings the instrument to a safe and stable state

#### property status

Requests and returns the status byte and Master Summary Status bit.

# 7.20.9 Keithley 2600 SourceMeter

```
class pymeasure.instruments.keithley.Keithley2600(adapter, **kwargs)
```

Bases: pymeasure.instruments.instrument.Instrument

Represents the Keithley 2600 series (channel A and B) SourceMeter

# check\_errors()

Logs any system errors reported by the instrument.

#### property complete

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

Returns a tuple of an error code and message from a single error.

#### property id

Requests and returns the identification of the instrument.

# property options

Requests and returns the device options installed.

# reset()

Resets the instrument.

# shutdown()

Brings the instrument to a safe and stable state

#### property status

Requests and returns the status byte and Master Summary Status bit.

# 7.21 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*.

# 7.21.1 Keysight DSOX1102G Oscilloscope

```
class pymeasure.instruments.keysight.KeysightDSOX1102G(adapter, **kwargs)
```

```
Bases: pymeasure.instruments.instrument.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".

# ask(command)

Writes the command to the instrument through the adapter and returns the read response.

**Parameters command** – command string to be sent to the instrument

#### autoscale()

Autoscale displayed channels.

#### check errors()

Read all errors from the instrument.

**Returns** list of error entries

#### clear()

Clears the instrument status byte

#### clear\_status()

Clear device status.

# property complete

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.

Returns a property for the class based on the supplied commands. This property may be set and read from the instrument.

#### **Parameters**

- **get\_command** A string command that asks for the value
- **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
- **get\_process** A function that take a value and allows processing before value mapping, returning the processed value

7.21. Keysight 159

- **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
- **check\_get\_errors** Toggles checking errors after getting

#### 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 less 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.

#### property id

Requests and returns the identification of the instrument.

Returns 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 take a value and allows processing before value mapping, returning the processed value
- **command\_process** A function that take a command and allows processing before executing the command, for both getting and setting
- **check\_get\_errors** Toggles checking errors after getting

### property options

Requests and returns the device options installed.

# read()

Reads from the instrument through the adapter and returns the response.

#### reset()

Resets the instrument.

#### run()

Starts repetitive acquisitions. This is the same as pressing the Run key on the front panel.

Returns 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

#### 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

Requests and returns 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 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

7.21. Keysight 161

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.

# values(command, \*\*kwargs)

Reads a set of values from the instrument through the adapter, passing on any key-word arguments.

# 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".

#### write(command)

Writes the command to the instrument through the adapter.

**Parameters command** – command string to be sent to the instrument

# 7.21.2 Keysight N5767A Power Supply

# class pymeasure.instruments.keysight.KeysightN5767A(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Keysight N5767A Power supply interface for interacting with the instrument.

# check\_errors()

Read all errors from the instrument.

**Returns** list of error entries

# property complete

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

Reads a setting current in Amps.

# property current\_range

A floating point property that controls the DC current range in Amps, which can take values from 0 to 25 A. Auto-range is disabled when this property is set.

# disable()

Disables the flow of current.

#### enable()

Enables the flow of current.

# property id

Requests and returns the identification of the instrument.

# is\_enabled()

Returns True if the current supply is enabled.

# property options

Requests and returns the device options installed.

# reset()

Resets the instrument.

# shutdown()

Brings the instrument to a safe and stable state

#### property status

Requests and returns the status byte and Master Summary Status bit.

# property voltage

Reads a DC voltage measurement in Volts.

7.21. Keysight 163

#### property voltage\_range

A floating point property that controls the DC voltage range in Volts, which can take values from 0 to 60 V. Auto-range is disabled when this property is set.

# 7.22 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*.

# 7.22.1 Lake Shore Adapters

class pymeasure.instruments.lakeshore.LakeShoreUSBAdapter(port)

Bases: pymeasure.adapters.serial.SerialAdapter

Provides a SerialAdapter with the specific baudrate, timeout, parity, and byte size for LakeShore USB communication.

Initiates the adapter to open serial communcation over the supplied port.

**Parameters port** – A string representing the serial port

**\_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

ask(command)

Writes the command to the instrument and returns the resulting ASCII response

**Parameters command** – SCPI command string to be sent to the instrument

**Returns** String ASCII response of the instrument

**binary\_values**(*command*, *header\_bytes*=0, *dtype*=<*class 'numpy.float32'*>)
Returns a numpy array from a query for binary data

#### **Parameters**

- command SCPI command to be sent to the instrument
- **header\_bytes** Integer number of bytes to ignore in header
- dtype The NumPy data type to format the values with

Returns NumPy array of values

read()

Reads until the buffer is empty and returns the resulting ASCII response

**Returns** String ASCII response of the instrument.

**values**(command, separator=', ', cast=<class 'float'>, preprocess\_reply=None)

Writes a command to the instrument and returns a list of formatted values from the result

#### **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. If not specified, the Adapter default is used if available, otherwise no preprocessing is done.

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

#### write(command)

Overwrites the *SerialAdapter.write* method to automatically append a Unix-style linebreak at the end of the command.

**Parameters** command – SCPI command string to be sent to the instrument

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

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

#### **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

# 7.22.2 Lake Shore 331 Temperature Controller

```
class pymeasure.instruments.lakeshore.LakeShore331(adapter, **kwargs)
    Bases: pymeasure.instruments.instrument.Instrument
```

Represents the Lake Shore 331 Temperature Controller and provides a high-level interface for interacting with the instrument.

```
controller = LakeShore331("GPIB::1")

print(controller.setpoint_1)  # Print the current setpoint for loop 1
controller.setpoint_1 = 50  # Change the setpoint to 50 K
controller.heater_range = 'low'  # Change the heater range to Low
controller.wait_for_temperature()  # Wait for the temperature to stabilize
print(controller.temperature_A)  # Print the temperature at sensor A
```

# disable\_heater()

Turns the heater\_range to off to disable the heater.

#### property heater\_range

A string property that controls the heater range, which can take the values: off, low, medium, and high. These values correlate to 0, 0.5, 5 and 50 W respectively.

# property setpoint\_1

A floating point property that controls the setpoint temperature in Kelvin for Loop 1.

#### property setpoint\_2

A floating point property that controls the setpoint temperature in Kelvin for Loop 2.

# property temperature\_A

Reads the temperature of the sensor A in Kelvin.

# property temperature\_B

Reads the temperature of the sensor B in Kelvin.

Blocks the program, waiting for the temperature to reach the setpoint within the accuracy (%), checking this each interval time in seconds.

#### **Parameters**

- accuracy An acceptable percentage deviation between the setpoint and temperature
- interval A time in seconds that controls the refresh rate
- **sensor** The desired sensor to read, either A or B
- **setpoint** The desired setpoint loop to read, either 1 or 2
- 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

# 7.22.3 Lake Shore 425 Gaussmeter

# class pymeasure.instruments.lakeshore.LakeShore425(port)

Bases: pymeasure.instruments.instrument.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

Returns 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 range

A floating point property that controls the field range in units of Gauss, which can take the values 35, 350, 3500, and 35,000 G.

# property unit

A string property that controls the units of the instrument, which can take the values of G, T, Oe, or A/m.

# zero\_probe()

Initiates the zero field sequence to calibrate the probe

# 7.23 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*.

# 7.23.1 ESP 300 Motion Controller

# class pymeasure.instruments.newport.ESP300(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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 availble axes. Axes are controlled through an Axis class.

# property axes

A list of the Axis objects that are present.

#### clear\_errors()

Clears the error messages by checking until a 0 code is recived.

# disable()

Disables all of the axes associated with this controller.

#### enable()

Enables all of the axes associated with this controller.

#### property error

Reads an error code from the motion controller.

# property errors

Returns 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.

7.23. Newport 167

#### 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: enconder 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.

#### 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.

# 7.24 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*.

# 7.24.1 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**

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: pymeasure.instruments.ni.virtualbench.VirtualBench.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-xxxxxxx/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/dig/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: pymeasure.instruments.ni.virtualbench.VirtualBench.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 dmm\_input\_resistance(int or str)-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**

- function index or name (dmm\_function:DMM) -
  - '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: pymeasure.instruments.ni.virtualbench.VirtualBench.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: pymeasure.instruments.ni.virtualbench.VirtualBench.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\_channel**(channel, enable\_channel, vertical\_range, vertical\_offset, probe\_attenuation, vertical\_coupling)

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')

**configure\_analog\_channel\_characteristics**(*channel*, *input\_impedance*, *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', 'LESS\_THAN\_LOWER\_LIMIT',
  'INSIDE\_LIMITS' or 'OUTSIDE\_LIMITS')
- 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: pymeasure.instruments.ni.virtualbench.VirtualBench.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

 $\begin{center} \textbf{Parameters timestamp} \ (\textit{pyvb.Timestamp}) - Virtual Bench \ timestamp \end{center}$ 

**Returns** Timestamp as DateTime object

Return type DateTime

# 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.  $\frac{\text{dig}}{0:2'} \rightarrow \frac{\text{dig}}{0}$ ,  $\frac{\text{dig}}{1}$ ,  $\frac{\text{dig}}{2'}$ , 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.

# 7.25 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*.

# 7.25.1 Oxford Instruments Intelligent Temperature Controller 503

Bases: pymeasure.instruments.instrument.Instrument

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
```

(continues on next page)

(continued from previous page)

### property auto\_pid

A boolean property that sets the Auto-PID mode on (True) or off (False).

#### property control\_mode

A string property that sets the ITC in LOCAL or REMOTE and LOCKES, or UNLOCKES, the LOC/REM button. Allowed values are: LL: LOCAL & LOCKED RL: REMOTE & LOCKED LU: LOCAL & UNLOCKED RU: REMOTE & UNLOCKED.

#### 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 heater

A floating point property that sets the required heater output when in manual mode. The parameter is expressed as a percentage of the maximum voltage. 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: MANUAL: HEATER MANUAL, GAS MANUAL AM: HEATER AUTO, GAS MANUAL MA: HEATER MANUAL, GAS AUTO AUTO: HEATER AUTO, GAS AUTO.

### 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 sweep\_status

An integer property that sets the sweep status. Values are: 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 sets values in the sweep table. Relies on the xpointer and ypointer to point at the location in the table that is to be set.

#### 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.

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 set to zero, 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.
- max\_comm\_errors The maximum number of communication errors that are allowed before the wait is stopped. if set to None (default), no maximum will be used.

#### property xpointer

An integer property to set pointers into tables for loading and examining values in the table. For programming the sweep table values from 1 to 16 are allowed, corresponding to the maximum number of steps.

### property ypointer

An integer property to set pointers into tables for loading and examining values in the table. For programming the sweep table the allowed values are: 1: Setpoint temperature, 2: Sweep-time to set-point, 3: Hold-time at set-point.

# 7.26 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*.

# 7.26.1 Parker GV6 Servo Motor Controller

### class pymeasure.instruments.parker.ParkerGV6(port)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Parker Gemini GV6 Servo Motor Controller and provides a high-level interface for interacting with the instrument

### property angle

Returns the angle in degrees based on the position and whether relative or absolute positioning is enabled, returning None on error

### property angle\_error

Returns the angle error in degrees based on the position error, or returns None on error

### disable()

Disables the motor from moving

#### echo(enable=False)

Enables (True) or disables (False) the echoing of all commands that are sent to the instrument

#### enable()

Enables the motor to move

### is\_moving()

Returns True if the motor is currently moving

#### kill()

Stops the motor

#### move()

Initiates the motor to move to the setpoint

### property position

Returns an integer number of counts that correspond to the angular position where 1 revolution equals 4000 counts

### property position\_error

Returns the error in the number of counts that corresponds to the error in the angular position where 1 revolution equals 4000 counts

#### read()

Overwrites the Instrument.read command to provide the correct functionality

### reset()

Resets the motor controller while blocking and (CAUTION) resets the absolute position value of the motor

### set\_defaults()

Sets up the default values for the motor, which is run upon construction

### set\_hardware\_limits(positive=True, negative=True)

Enables (True) or disables (False) the hardware limits for the motor

#### set\_software\_limits(positive, negative)

Sets the software limits for motion based on the count unit where 4000 counts is 1 revolution

### property status

Returns a list of the motor status in readable format

# stop()

Stops the motor during movement

7.26. Parker 183

#### use\_absolute\_position()

Sets the motor to accept setpoints from an absolute zero position

#### use\_relative\_position()

Sets the motor to accept setpoints that are relative to the last position

#### write(command)

Overwrites the Insturment.write command to provide the correct line break syntax

# 7.27 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*.

# 7.27.1 Pendulum CNT91 frequency counter

### class pymeasure.instruments.pendulum.cnt91.CNT91(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents a Pendulum CNT-91 frequency counter.

#### property batch\_size

Maximum number of buffer entries that can be transmitted at once.

### buffer\_frequency\_time\_series(channel, n\_samples, sample\_rate, trigger\_source=None)

Record a time series to the buffer and read it out after completion.

#### **Parameters**

- channel Channel that should be used
- **n\_samples** The number of samples
- sample\_rate Sample rate in Hz
- trigger\_source Optionally specify a trigger source to start the measurement

### configure\_frequency\_array\_measurement(n\_samples, channel)

Configure the counter for an array of measurements.

#### **Parameters**

- **n\_samples** The number of samples
- **channel** Measurment channel (A, B, C, E, INTREF)

# property continuous

Controls whether to perform continuous measurements.

### property external\_arming\_start\_slope

Set slope for the start arming condition.

#### property external\_start\_arming\_source

Select arming input or switch off the start arming function. Options are 'A', 'B' and 'E' (rear). 'IMM' turns trigger off.

# property format

Reponse format (ASCII or REAL).

### property interpolator\_autocalibrated

Controls if interpolators should be calibrated automatically.

#### property measurement\_time

Gate time for one measurement in s.

```
read_buffer(expected_length=0)
```

Read out the entire buffer.

**Parameters expected\_length** – The expected length of the buffer. If more data is read, values at the end are removed. Defaults to 0, which means that the entire buffer is returned independent of its length.

**Returns** Frequency values from the buffer.

# 7.28 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*.

# 7.28.1 Razorbill RP100 custrom power supply for Razorbill Instrums stress & strain cells

```
class pymeasure.instruments.razorbill.razorbillRP100(adapter, **kwargs)
```

Bases: pymeasure.instruments.instrument.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

Returns the current in amps present at the front panel output of channel 1

### property contact\_current\_2

Returns the current in amps present at the front panel output of channel 2

#### property contact\_voltage\_1

Returns the Voltage in volts present at the front panel output of channel 1

### property contact\_voltage\_2

Returns the Voltage in volts present at the front panel output of channel 2

### property instant\_voltage\_1

Returns the instantaneous output of source one in volts

#### property instant\_voltage\_2

Returns the instanteneous output of source two in volts

### property output\_1

Turns output of channel 1 on or off

### property output\_2

Turns output of channel 2 on or off

#### property slew\_rate\_1

Sets or queries the source slew rate in volts/sec of channel 1

7.28. Razorbill 185

#### property slew\_rate\_2

Sets or queries the source slew rate in volts/sec of channel 2

### property voltage\_1

Sets or queries the output voltage of channel 1

# property voltage\_2

Sets or queries the output voltage of channel 2

# 7.29 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*.

### 7.29.1 R&S SFM TV test transmitter

class pymeasure.instruments.rohdeschwarz.sfm.SFM(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Rohde&Schwarz SFM TV test transmitter interface for interacting with the instrument.

**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 implmented.

### 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 infomation about the hardware modules installed in the unit

# property beeper\_enabled

A bool property that controls the beeper status,

7.29. Rohde & Schwarz 187

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:

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	Continous wave mode
FIXED	fixed frequency mode
CHSW	Channel sweep
RFSW	Frequency sweep

**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	continous mode	+10 dBm
LOWD	low distortion mode	+0 dBm

Contiuous 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

### 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 artifical bit error rate to be applied

### property nicam\_bit\_error\_rate

A float property that controls the artifical 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

7.29. Rohde & Schwarz 191

#### 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	Continous 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

193

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:

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

7.29. Rohde & Schwarz

• 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
EVEN	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 infomation 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

#### property subsystem\_info

A String property containing infomation 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

7.29. Rohde & Schwarz 195

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

refere 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

# 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.

# 7.29.2 R&S FSL spectrum analyzer

### Connecting to the instrument 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:

```
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()
m1.to_peak()
m1.zoom('20 MHz')

# take data from the zoomed-in region
fsl.single_sweep()
x, y = fsl.read_trace()
```

```
class pymeasure.instruments.rohdeschwarz.fsl.FSL(resourceName, **kwargs)
```

Bases: pymeasure.instruments.instrument.Instrument

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.).

### property attenuation

Attenuation in dB.

### continue\_single\_sweep()

Continue with single sweep with synchronization.

#### property continuous\_sweep

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

Center frequency in Hz.

### property freq\_span

Frequency span in Hz.

#### property freq\_start

Start frequency in Hz.

### property freq\_stop

Stop frequency in Hz.

```
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]].

#### property res\_bandwidth

Resolution bandwidth in Hz. Can be set to 'AUTO'

#### single\_sweep()

Perform a single sweep with synchronization.

### property sweep\_time

Sweep time in s. Can be set to 'AUTO'.

#### property trace\_mode

Trace mode ('WRIT', 'MAXH', 'MINH', 'AVER' or 'VIEW')

### property video\_bandwidth

Video bandwidth in Hz. Can be set to 'AUTO'

# 7.30 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*.

# 7.30.1 DSP 7265 Lock-in Amplifier

class pymeasure.instruments.signalrecovery.DSP7265(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

This is the class for the DSP 7265 lockin amplifier

#### property adc1

Reads the input value of ADC1 in Volts

# property adc2

Reads the input value of ADC2 in Volts

buffer\_to\_float(buffer\_data, sensitivity=None, sensitivity2=None, raise\_error=True)

Method that converts the 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, magnitude2 and noise2 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 elements in the 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) Boolean that 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.

### property curve\_buffer\_bits

An integer property that controls which data outputs are stored in the curve buffer. Valid values are values between 1 and 65,535 (or 2,097,151 in dual reference mode).

#### property curve\_buffer\_interval

An integer property that controls Sets the time interval between successive points being acquired in the curve buffer. The time interval is specified 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

An integer property that controls the length of the curve buffer. Valid values are values 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

A property that represents the status of the curve buffer acquisition with four values: the first value represents the status with 5 possibilities (0: no activity, 1: acquisition via TD command running, 2: acquisition by TDC command running, 5: acquisition via TD command halted, 6: acquisition bia TDC command halted); the second value is the number of sweeps that is acquired; the third value is the decimal representation of the status byte (the same response as the ST command; the fourth value is the number of points acquired in the curve buffer.

### property dac1

A floating point property that represents the output value on DAC1 in Volts. This property can be set.

### property dac2

A floating point property that represents the output value on DAC2 in Volts. This property can be set.

#### property dac3

A floating point property that represents the output value on DAC3 in Volts. This property can be set.

### property dac4

A floating point property that represents the output value on DAC4 in Volts. This property can be set.

### property frequency

A floating point property that represents the lock-in frequency in Hz. This property can be set.

#### get\_buffer(quantity=None, convert\_to\_float=True, wait\_for\_buffer=True)

Method that 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

An integer property that represents the reference harmonic mode control, taking values from 1 to 65535. This property can be set.

#### property id

Reads the instrument identification

### property imode

Property that controls the voltage/current mode. can be '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

Reads the log ratio output, defined as log(X/ADC1)

### property mag

Reads the magnitude in Volts

#### property phase

Reads the phase in degrees

### property ratio

Reads the ratio output, defined as X/ADC1

#### property reference

Controls the oscillator reference. Can be "internal", "external rear" or "external front"

#### property reference\_phase

A floating point property that represents the reference harmonic phase in degrees. This property can be set.

### property sensitivity

A floating point property that controls the sensitivity range in Volts (for voltage mode) or Amps (for current modes). When in Volts it takes discrete values from 2 nV to 1 V. When in Amps it takes discrete values from 2 fA to 1  $\mu$ A (for normal current mode) or up to 10 nA (for low noise current mode). This property can be set.

#### setDifferentialMode(lineFiltering=True)

Sets lockin to differential mode, measuring A-B

#### set\_buffer(points, quantities=None, interval=0.01)

Method that prepares the curve buffer for a measurement.

#### **Parameters**

- **points** (*int*) Number of points to be recorded in the curve buffer
- quantities (1ist) 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.

#### shutdown()

Brings the instrument to a safe and stable state

#### property slope

A integer property that controls the filter slope in dB/octave, which can take the values 6, 12, 18, or 24 dB/octave. This property can be set.

#### 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 time\_constant

A floating point property that controls the time constant in seconds, which takes values from 10 microseconds to 50,000 seconds. This property can be set.

### property voltage

A floating point property that represents the voltage in Volts. This property can be set.

### wait\_for\_buffer(timeout=None, delay=0.1)

Method that waits until the curve buffer is filled

### property x

Reads the X value in Volts

#### property xy

Reads both the X and Y values in Volts

#### property y

Reads the Y value in Volts

# 7.31 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*.

# 7.31.1 SR510 Lock-in Amplifier

# class pymeasure.instruments.srs.SR510(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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.

# 7.31.2 SR570 Lock-in Amplifier

### class pymeasure.instruments.srs.SR570(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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

# 7.31.3 SR830 Lock-in Amplifier

### class pymeasure.instruments.srs.SR830(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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.

#### 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.

#### 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 0: unused bit 1: backup error bit 2: RAM error bit 3: unused bit 4: ROM error bit 5: GPIB error bit 6: DSP error bit 7: Math error

### 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 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)

Aquires the 32 bit floating point data through binary transfer

#### get\_scaling(channel)

Returns the offset precent and the exapnsion 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 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 0: Input/Amplifier overload bit 1: Time constant filter overload bit 2: Output overload bit 3: Reference unlock bit 4: Detection frequency range switched bit 5: Time constant changed indirectly bit 6: Data storage triggered bit 7: unused

#### 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)

### 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 sample\_frequency

Gets the sample frequency in Hz

### 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 precent (-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)

#### 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.

# 7.31.4 SR860 Lock-in Amplifier

### class pymeasure.instruments.srs.SR860(resourceName, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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 +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

A string property that represents the synchronous filter. This property can be set. Allowed values are: ['Off', 'On']

### 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 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 that sets 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\_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\_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\_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  $strip\ chart\ history(i=3)$ , half  $strip\ chart\ h$ 

#### 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 sensitvity

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 sine\_amplitudepreset1

A floating point property that represents the preset sine out amplitude, for the A1 preset button. This property can be set.

### property sine\_amplitudepreset2

A floating point property that represents the preset sine out amplitude, for the A2 preset button. This property can be set.

### property sine\_amplitudepreset3

A floating point property that represents the preset sine out amplitude, for the A3 preset button. This property can be set.

### property sine\_amplitudepreset4

A floating point property that represents 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:

j enumeration parameter j enumeration parameter

0 X X output 9 YNOise Ynoise 1 Y Youtput 10 OUT1 Aux Out1 2 R R output 11 OUT2 Aux Out2 3 THeta output 12 PHAse Reference Phase 4 IN1 Aux In1 13 SAMp Sine Out Amplitude 5 IN2 Aux In2 14 LEVel DC Level 6 IN3 Aux In3 15 FInt Int. Ref. Frequency 7 IN4 Aux In4 16 FExt Ext. Ref. Frequency 8 XNOise Xnoise

#### **Parameters**

- val1 parameter enumeration/index
- val2 parameter enumeration/index
- val3 parameter enumeration/index (optional)

**Defaults:** val1 = "X" val2 = "Y" val3 = None

## 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

## 7.32 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*.

## 7.32.1 TDS2000 Oscilloscope

```
class pymeasure.instruments.tektronix.TDS2000(resourceName, **kwargs)
```

Bases: pymeasure.instruments.instrument.Instrument

Represents the Tektronix TDS 2000 Oscilloscope and provides a high-level for interacting with the instrument

## 7.32.2 AFG3152C Arbitrary function generator

#### class pymeasure.instruments.tektronix.AFG3152C(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.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

7.32. Tektronix 213

## 7.33 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*.

## 7.33.1 Thorlabs PM100USB Powermeter

```
class pymeasure.instruments.thorlabs.ThorlabsPM100USB(adapter, **kwargs)
    Bases: pymeasure.instruments.instrument.Instrument
    Represents Thorlabs PM100USB powermeter.
    property energy
        Energy in J.
    property power
        Power in W.
    property wavelength
        Wavelength in nm.
    property wavelength_max
        Maximum wavelength, in nm
```

#### ivianimum waverengun,

property wavelength\_min

Minimum wavelength, in nm

## 7.33.2 Thorlabs Pro 8000 modular laser driver

```
class pymeasure.instruments.thorlabs.ThorlabsPro8000(resourceName, **kwargs)
     Bases: pymeasure.instruments.instrument.Instrument
     Represents Thorlabs Pro 8000 modular laser driver
     property LDCCurrent
          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
           Set TEC temperature
     property TEDStatus
           Set TEC status. Allowed values are: ['ON', 'OFF']
     property slot
           Slot selection. Allowed values are: range(1, 9)
```

# 7.34 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*.

## 7.34.1 Yokogawa 7651 Programmable Supply

class pymeasure.instruments.yokogawa.Yokogawa7651(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Yokogawa 7651 Programmable DC Source and provides a high-level for interacting with the instrument.

```
yoko = Yokogawa7651("GPIB::1")

yoko.apply_current()  # Sets up to source current
yoko.source_current_range = 10e-3  # Sets the current range to 10 mA
yoko.compliance_voltage = 10  # Sets the compliance voltage to 10 V
yoko.source_current = 0  # Sets the source current to 0 mA

yoko.enable_source()  # Enables the current output
yoko.ramp_to_current(5e-3)  # Ramps the current to 5 mA

yoko.shutdown()  # Ramps the current to 0 mA and disables output
```

```
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 <code>source\_voltage\_range</code> and <code>compliance\_current</code> properties.

### property compliance\_current

A floating point property that sets the compliance current in Amps, which can take values from 5 to  $120 \, \text{mA}$ 

## property compliance\_voltage

A floating point property that sets 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

Returns 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

7.34. Yokogawa 215

• 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

A floating point property that controls the source current in Amps, if that mode is active.

#### property source\_current\_range

A floating point property that sets the current voltage range in Amps, which can take values: 1 mA, 10 mA, and 100 mA. Currents are truncted to an appropriate value if needed.

#### property source\_enabled

Reads 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

A string property that controls 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

A floating point property that controls the source voltage in Volts, if that mode is active.

## property source\_voltage\_range

A floating point property that sets the source voltage range in Volts, which can take values: 10 mV, 100 mV, 1 V, 10 V, and 30 V. Voltages are truncted to an appropriate value if needed.

## 7.34.2 Yokogawa GS200 Source

#### class pymeasure.instruments.yokogawa.YokogawaGS200(adapter, \*\*kwargs)

Bases: pymeasure.instruments.instrument.Instrument

Represents the Yokogawa GS200 source and provides a high-level interface for interacting with the instrument.

#### property current\_limit

Floating point number that controls 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

A boolean property that controls whether the source is enabled, takes values True or False.

#### property source\_level

Floating point number that controls the output level, either a voltage or a current, depending on the source mode.

#### property source\_mode

String property that controls the source mode. Can be either 'current' or 'voltage'.

## property source\_range

Floating point number that controls the range (either in voltage or current) of the output. "Range" refers to the maximum source level.

## 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

Floating point number that controls 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

7.34. Yokogawa 217

**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.

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.

```
cd /path/for/code/pymeasure pip install -e .
```

This will allow you to edit the files of PyMeasure and see the changes reflected. 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!

# 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, its helpful to start a pull-request (PR) on 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 after your first commit.

Start a pull-request on the PyMeasure GitHub page.

Your pull-request will be merged by the PyMeasure maintainers once it meets the coding standards and passes unit tests. You will notice that your pull-request is automatically checked with the unit tests.

# 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.

```
python setup.py test
```

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 221

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.

**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. This section 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. For a new manufacturer, the manufacturer should also be added to pymeasure/pymeasure/instruments/\_\_init\_\_.py.

## 10.1.2 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) 2013-2021 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:

```
class Extreme5000(Instrument):
    """ Represents the imaginary Extreme 5000 instrument.

def __init__(self, resourceName, **kwargs):
    super().__init__(
        resourceName,
        "Extreme 5000",
        **kwargs
    )
```

Make sure to include the PyMeasure license to each file, and add yourself as an author to the AUTHORS.txt file.

In principle you are free to write any functions 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 command instead of calling the adapter directly.

In practice, we have developed a number of convenience functions for making instruments easy to write and maintain. The following sections detail these conveniences and are highly encouraged.

# 10.3 Writing properties

In PyMeasure, Python properties are the preferred method for dealing with variables that are read or set. PyMeasure comes with two convenience functions for making properties for classes. The *Instrument.measurement* function returns a property that issues a GPIB/SCPI requests when the value is used. For example, if our "Extreme 5000" has the \*IDN? command we can write the following property to be added above the def \_\_init\_\_ line in our above example class, or added to the class after the fact as in the code here:

```
Extreme5000.id = Instrument.measurement(
    "*IDN?", """ Reads the instrument identification """
)
```

You will notice that a documentation string is required, and should be descriptive and specific.

When we use this property we will get the identification information.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.id  # Reads "*IDN?"
'Extreme 5000 identification from instrument'
```

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",
    """ A floating point property that controls the voltage
    in Volts. This property can be set.
    """
)
```

You will notice that we use the Python string format %g to pass through the floating point.

We can use this property to set the voltage to 100 mV, which will execute the command and then request the current voltage.

```
>>> extreme = Extreme5000("GPIB::1")
>>> extreme.voltage = 0.1  # Executes ":VOLT 0.1"
>>> extreme.voltage  # Reads ":VOLT?"
0.1
```

Using Instrument.control and Instrument.measurement functions, you can create a number of properties for basic measurements and controls.

The *Instrument.control* function can be used with multiple values at once, 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",
    """ A floating point property that simultaneously controls the voltage
    in Volts and the frequency in Hertz. This property can be 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]
```

The next section details additional features of *Instrument.control* that allow you to write properties that cover specific ranges, or 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.

# 10.4 Advanced properties

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* function also provide means to achieve this.

## 10.4.1 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",
    """ A floating point property that controls the voltage
    in Volts, from -1 to 1 V. This property can be set. """,
    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",
    """ A floating point property that controls the voltage
    in Volts, from -1 to 1 V. Invalid voltages are truncated.
```

(continues on next page)

(continued from previous page)

```
This property can be set. """,
  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
```

### 10.4.2 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.

```
Extreme5000.voltage = Instrument.control(
    ":RANG?", ":RANG %g",
    """ A floating point property that controls the voltage
    range in Volts. This property can be set.
    """,
    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
```

## 10.4.3 Using maps

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 *Instrument.control* 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",
    """ A floating point property that controls the voltage
    range in Volts, which takes values of 10 mV, 100 mV and 1 V.
    This property can be set. """,
```

(continues on next page)

(continued from previous page)

```
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",
    """ A floating point property that controls the voltage
    range in Volts, which takes values of 10 mV, 100 mV and 1 V.
    This property can be set. """,
    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",
    """ A string property that controls the measurement channel,
    which can take the values X, Y, or 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.

## 10.4.4 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",
    """ A floating point property that takes the measurement current in A
    """,
    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
```

## 10.4.5 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",
    """ A floating point property representing the measurement current in A
    """,
    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
```

*get\_process* can also be used to perform string processing. Let's say your instrument returns a value with its unit which has to be removed. This could be achieved by the following code:

```
Extreme5000.capacity = Instrument.measurement(
    ":CAP?",
    """ A measurement returning a capacity in nF in the format '<cap> nF'
    """,
    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?",
    """ A measurement returning a capacity in nF in the format '<cap> nF'
    """,
    preprocess_reply=lambda v: v.replace('nF', '')
    # notice how we don't need to cast to float anymore
)
```

The real purpose of *preprocess\_reply* is, however, for instruments where many/all properties need similar reply processing. *preprocess\_reply* can be applied to all *Instrument.control* or *Instrument.measurement* properties, for example if all quantities are returned with a unit as in the example above. To avoid running into troubles for other properties this *preprocess\_reply* should be clever enough to skip the processing in case it is not appropriate, for example if some identification string is returned. Typically this can be achieved by regular expression matching. In case of no match the reply is returned unchanged:

```
import re
_{reg\_value} = re.compile(r''([-+]?[0-9]*\.?[0-9]+)\s+\w+")
def extract_value(reply):
    """ extract numerical value from reply. If none can be found the reply
   is returned unchanged.
    :param reply: reply string
    :returns: string with only the numerical value
   r = _reg_value.search(reply)
   if r:
        return r.groups()[0]
   else:
       return reply
class Extreme5001(Instrument):
    """ Represents the imaginary Extreme 5001 instrument. This instrument
    sends numerical values including their units in an format "<value>
    <unit>".
    capacity = Instrument.measurement(
        ":CAP?".
        """ A measurement returning a capacity in nF in the format '<cap> nF'
```

(continues on next page)

(continued from previous page)

```
)
voltage = Instrument.measurement(
    ": VOLT?",
    """ A measurement returning a voltage in V in the format '<volt> \mbox{V}\mbox{'}
)
id = Instrument.measurement(
    "*idn?",
    """ The identification of the instrument.
)
def __init__(self, resourceName, **kwargs):
    super().__init__(
        resourceName,
        "Extreme 5000",
        preprocess_reply=extract_value,
        **kwargs,
    )
```

In cases where the general *preprocess\_reply* function should not run it can be also overwritten in the property definition:

```
Extreme5001.channel = Instrument.control(
    ":CHAN?", ":CHAN %d",
    """ A string property that controls the measurement channel,
    which can take the values X, Y, or Z.
    """,
    validator=strict_discrete_set,
    values=[1,2,3],
    preprocess_reply=lambda v: v,
)
```

Using a combination of the decribed abilities also complex communication schemes can be achieved.

**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 seperate words. CamelCase should only be used for class names, unless working with Qt, where its use is common.

There are no plans to support type hinting in PyMeasure code. This adds a lot of additional code to manage, without a clear advantage for this project. Type documentation should be placed in the docstring where not clear from the variable name.

## 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.

# 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 *Advanced properties* section, these types of properties are prefered.

**CHAPTER** 

## **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

## PyMeasure Documentation, Release 0.9.0

Jonathan Larochelle

Dominic Caron

Mathieu Plante

Michele Sardo

Steven Siegl

Benjamin Klebel-Knobloch

Demetra Adrahtas

Dan McDonald

Hud Wahab

Nicola Corna

Robert Eckelmann

Sam Condon

Andreas Maeder

Bastian Leykauf

**CHAPTER** 

## **THIRTEEN**

## **LICENSE**

## Copyright (c) 2013-2021 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.

240 Chapter 13. License

## **PYTHON MODULE INDEX**

```
р
                                               pymeasure.instruments.newport, 167
                                               pymeasure.instruments.ni, 168
pymeasure.display.browser, 49
                                               pymeasure.instruments.oxfordinstruments, 180
pymeasure.display.curves, 49
                                               pymeasure.instruments.parker, 182
pymeasure.display.inputs, 50
                                               pymeasure.instruments.pendulum, 184
pymeasure.display.listeners, 52
                                               pymeasure.instruments.razorbill, 185
pymeasure.display.log, 52
                                               pymeasure.instruments.rohdeschwarz, 186
pymeasure.display.manager, 52
                                               pymeasure.instruments.signalrecovery, 200
pymeasure.display.plotter, 53
                                               pymeasure.instruments.srs, 203
pymeasure.display.thread, 54
                                               pymeasure.instruments.tektronix, 213
pymeasure.display.widgets, 54
                                               pymeasure.instruments.thorlabs, 213
pymeasure.display.windows, 57
                                               pymeasure.instruments.validators, 63
pymeasure.experiment.experiment, 41
                                               pymeasure.instruments.yokogawa, 214
pymeasure.experiment.listeners, 42
pymeasure.experiment.parameters, 44
pymeasure.experiment.procedure, 43
pymeasure.experiment.results, 47
pymeasure.experiment.workers, 46
pymeasure.instruments, 59
pymeasure.instruments.advantest, 65
pymeasure.instruments.advantest.advantestR3767CG,
pymeasure.instruments.agilent, 66
pymeasure.instruments.agilent.agilent4156,76
pymeasure.instruments.agilent.agilentB1500,
        101
pymeasure.instruments.ametek, 103
pymeasure.instruments.ami, 105
pymeasure.instruments.anaheimautomation, 107
pymeasure.instruments.anapico, 109
pymeasure.instruments.anritsu, 109
pymeasure.instruments.attocube, 112
pymeasure.instruments.bkprecision, 115
pymeasure.instruments.comedi, 65
pymeasure.instruments.danfysik, 116
pymeasure.instruments.deltaelektronika, 119
pymeasure.instruments.edwards, 120
pymeasure.instruments.fluke, 120
pymeasure.instruments.fwbell, 120
pymeasure.instruments.hp, 122
pymeasure.instruments.keithley, 126
pymeasure.instruments.keysight, 158
pymeasure.instruments.lakeshore, 164
```

242 Python Module Index

# **INDEX**

Symbols	method), 179		
call() (pymeasure.instruments.agilent.agilentB1500	Require_power_supply() (pymea-		
method), 100	sure.instruments.nt.virtuatbench.virtuatbench		
str() (pymeasure.instruments.agilent.agilentB1500.0	CustomIntEntithod), 179		
method), 101	acquire_reference() (pymeu-		
_format_binary_values() (pymea-	sure.instruments.keithley.Keithley2000		
sure.adapters.PrologixAdapter method),	method), 126		
35	acquisition_mode (pymea-		
_format_binary_values() (pymea-	sure.instruments.keysight.KeysightDSOX1102G		
sure.adapters.SerialAdapter method), 33	property), 159		
_format_binary_values() (pymea-	acquisition_type (pymea-		
sure.instruments.lakeshore.LakeShoreUSBAdapte method), 164	sure.instruments.keysight.KeysightDSOX1102G property), 159		
,, == :	active_connectors (pymea-		
A	sure.instruments.hp.HP3478A property),		
<pre>abort() (pymeasure.display.manager.Manager method),</pre>	123		
	Adapter (class in pymeasure.adapters), 31		
abort() (pymeasure.instruments.agilent.agilentB1500.Agi	adc1 (pymeasure.instruments.ametek.Ametek7270 prop-		
method), 94	erty), 104		
absolute nosition (nymag	adc1 (pymeasure.instruments.signalrecovery.DSP7265		
sure instruments analysimautomation DPSeriesM	property), 200 lotorController adc1 (pymeasure.instruments.srs.SR830 property), 205		
property), 107	adc1 (pymeasure.instruments.srs.SR830 property), 205		
1 1 27	adc1 (pymeasure.instruments.srs.SR860 property), 208		
sure instruments analysimautomation DPSeriesM	adc2 (pymeasure.instruments.ametek.Ametek7270 prop- lotorController erty), 104		
method), 107	erty), 104		
ac_current (pymeasure.instruments.agilent.AgilentE4980	adc2 (pymeasure.instruments.signalrecovery.DSP7265		
	Droberty), ZMO		
property), 71 ac_mode() (pymeasure.instruments.lakeshore.LakeShore4.	2adc2 (pymeasure.instruments.srs.SR830 property), 205		
method), 166	adc2 (pymeasure.instruments.srs.SR860 property), 208		
ac_voltage(pymeasure.instruments.agilent.AgilentE4980	adc3 (pymeasure.instruments.ametek.Ametek7270 prop-		
property), 71	erty), 104		
acquire_digital_input_output() (pymea-	adc3 (pymeasure.instruments.srs.SR830 property), 205		
sure.instruments.ni.virtualbench.VirtualBench	adc3 (pymeasure.instruments.srs.SR860 property), 208		
method), 179	adc4 (pymeasure.instruments.ametek.Ametek7270 prop-		
acquire_digital_multimeter() (pymea-	erty), 104		
sure.instruments.ni.virtualbench.VirtualBench	adc4 (pymeasure.instruments.srs.SR830 property), 205		
method), 179	adc4 (pymeasure.instruments.srs.SR860 property), 208		
acquire_function_generator() (pymea-	adc_auto_zero (pymea-		
sure.instruments.ni.virtualbench.VirtualBench	sure.instruments.agilent.agilentB1500.AgilentB1500		
method), 179	property), 95		
acquire_mixed_signal_oscilloscope() (pymea-	adc_averaging() (pymea-		
sure instruments ni virtualbench VirtualBench	sure.instruments.agilent.agilentB1500.AgilentB1500		

method), 95	amplitude_source (pymea-
adc_setup() (pymeasure.instruments.agilent.agilentB15 method), 95	- 4 ×
adc_type (pymeasure.instruments.agilent.agilentB1500.S	• •
property), 97	sure.instruments.agilent.Agilent33220A
ADCMode (class in pymea-	
sure.instruments.agilent.agilentB1500), 101	amplitude_unit (pymea-
ADCType (class in pymea-	
sure.instruments.agilent.agilentB1500), 101	erty), 85
add() (pymeasure.display.browser.Browser method), 49	· · · · · · · · · · · · · · · · · · ·
add_ramp_step() (pymea-	
sure.instruments.danfysik.Danfysik8500	122
method), 116	analysis (pymeasure.instruments.anritsu.AnritsuMS9710C
address (pymeasure.instruments.anaheimautomation.DF	
property), 107	analysis_result (pymea-
	**
AdvantestR3767CG (class in pymea- sure.instruments.advantest.advantestR3767CG)	
66	
	*
AFG3152C (class in pymeasure.instruments.tektronix), 213	property), 78
Agilent33220A (class in pymea-	
sure.instruments.agilent), 83	sure.instruments.attocube.anc300), 113
Agilent33500 (class in pymeasure.instruments.agilent), 85	angle (pymeasure.instruments.parker.ParkerGV6 property), 183
Agilent33521A (class in pymea-	<pre>angle_error(pymeasure.instruments.parker.ParkerGV6</pre>
sure.instruments.agilent), 88	property), 183
Agilent34410A (class in pymea-	AnritsuMG3692C (class in pymea-
sure.instruments.agilent), 72	sure.instruments.anritsu), 110
Agilent34450A (class in pymea-	AnritsuMS9710C (class in pymea-
sure.instruments.agilent), 73	sure.instruments.anritsu), 110
Agilent4156 (class in pymeasure.instruments.agilent.agilent4156), 76	<pre>aperture() (pymeasure.instruments.agilent.AgilentE4980</pre>
Agilent8257D (class in pymeasure.instruments.agilent),	**
66	method), 49
Agilent8722ES (class in pymea- sure.instruments.agilent), 69	
AgilentB1500 (class in pymea-	
sure.instruments.agilent.agilentB1500), 93	sure.instruments.keithley.Keithley2400
AgilentE4408B (class in pymea-	method), 134
sure.instruments.agilent), 70	apply_current() (pymea-
AgilentE4980 (class in pymeasure.instruments.agilent),	sure.instruments.keithley.Keithley2450
71	method), 140
Ametek7270 (class in pymeasure.instruments.ametek),	apply_current() (pymea-
104	sure.instruments.yokogawa.Yokogawa7651
AMI430 (class in pymeasure.instruments.ami), 105	method), 215
amplitude (pymeasure.instruments.agilent.Agilent33220 property), 83	A apply_voltage() (pymea- sure.instruments.keithley.Keithley2400
amplitude (pymeasure.instruments.agilent.Agilent33500	·
property), 85	apply_voltage() (pymea-
amplitude (pymeasure.instruments.hp.HP33120A prop-	sure.instruments.keithley.Keithley2450
erty), 122	method), 141
amplitude_depth (pymea-	apply_voltage() (pymea-
sure.instruments.agilent.Agilent8257D prop-	sure.instruments.keithley.Keithley6517B
erty), 66	method), 154

244 Index

apply_voltage() (pymea-	sure.instruments.keithley.Keithley2400 prop-
sure.instruments.yokogawa.Yokogawa7651	erty), 134
method), 215	auto_pid(pymeasure.instruments.oxfordinstruments.ITC503
APSIN12G (class in pymeasure.instruments.anapico), 109	property), 181
arb_advance (pymeasure.instruments.agilent.Agilent3350	
property), 85	sure.instruments.fwbell.FWBell5080 method),
arb_file (pymeasure.instruments.agilent.Agilent33500	121
property), 85	auto_range() (pymea-
arb_filter(pymeasure.instruments.agilent.Agilent33500 property), 85	sure.instruments.keithley.Keithley2000 method), 127
<pre>arb_srate(pymeasure.instruments.agilent.Agilent33500</pre>	auto_range() (pymea-
property), 85	sure.instruments.lakeshore.LakeShore425
arb_srate(pymeasure.instruments.agilent.Agilent33521A	
property), 88	auto_range_enabled (pymea-
ask() (pymeasure.adapters.Adapter method), 31	sure.instruments.hp.HP3478A property),
ask() (pymeasure.adapters.FakeAdapter method), 32	124
ask() (pymeasure.adapters.PrologixAdapter method), 35	<pre>auto_range_source()</pre>
ask() (pymeasure.adapters.SerialAdapter method), 33	sure.instruments.keithley.Keithley2400
ask() (pymeasure.adapters.TelnetAdapter method), 39	method), 134
ask() (pymeasure.adapters.VISAAdapter method), 37	<pre>auto_range_source()</pre>
ask() (pymeasure.adapters.VXI11Adapter method), 38	sure.instruments.keithley.Keithley2450
ask() (pymeasure.instruments.agilent.agilentB1500.SMU	method), 141
method), 97	<pre>auto_range_source()</pre>
ask() (pymeasure.instruments.anaheimautomation.DPSer	
method), 107	method), 154
ask() (pymeasure.instruments.attocube.adapters.Attocube	Cautrolestatappei) (pymea-
method), 112	sure. instruments. ni. virtual bench. Virtual Bench. Mixed Signal Oscillari, and the property of the propert
ask() (pymeasure.instruments.fwbell.FWBell5080	method), 174
method), 121	auto_zero (pymeasure.instruments.keithley.Keithley2400
ask() (pymeasure.instruments.Instrument method), 61	property), 134
ask() (pymeasure.instruments.keysight.KeysightDSOX110	2 <b>G</b> uto_zero_enabled ( <i>pymea-</i>
method), 159	sure.instruments.hp.HP3478A property),
ask() (pymeasure.instruments.lakeshore.LakeShoreUSBA	dapter 124
method), 164	AutoManual (class in pymea-
<pre>ask_raw() (pymeasure.adapters.VXI11Adapter method),</pre>	sure.instruments.agilent.agilentB1500), 102
38	autoscale() (pymeasure.instruments.keysight.KeysightDSOX1102G
ask_values() (pymeasure.adapters.VISAAdapter	method), 159
method), 37	<pre>aux_in_1 (pymeasure.instruments.srs.SR830 property),</pre>
attenuation(pymeasure.instruments.rohdeschwarz.fsl.Fs.	SL 205
property), 199	<pre>aux_in_1 (pymeasure.instruments.srs.SR860 property),</pre>
AttocubeConsoleAdapter (class in pymea-	208
sure.instruments.attocube.adapters), 112	<pre>aux_in_2 (pymeasure.instruments.srs.SR830 property),</pre>
AUTO (pymeasure.instruments.agilent.agilentB1500.ADCM	
attribute), 101	<pre>aux_in_2 (pymeasure.instruments.srs.SR860 property),</pre>
AUTO (pymeasure.instruments.agilent.agilentB1500.AutoM	anual 208
attribute), 102	<pre>aux_in_3 (pymeasure.instruments.srs.SR830 property),</pre>
AUTO (pymeasure.instruments.agilent.agilentB1500.Compl	iancePolari®06
attribute), 103	aux_in_3 (pymeasure.instruments.srs.SR860 property),
auto_calibration (pymea-	208
**	5@ux_in_4 (pymeasure.instruments.srs.SR830 property),
property), 94	206
<pre>auto_offset() (pymeasure.instruments.srs.SR830</pre>	aux_in_4 (pymeasure.instruments.srs.SR860 property),
method), 205	
meinoa), 203	208

Index 245

	206	beep()	(pymeasure.instruments.keithley.Kei	thley2400
aux_ou	t_1 (pymeasure.instruments.srs.SR860 property), 209	beep()	method), 134	thlav2450
alix Oli	t_2 (pymeasure.instruments.srs.SR830 property),	beeh()	(pymeasure.instruments.keithley.Kei method), 141	iniey2430
uun_ou	206	beep()	(pymeasure.instruments.keithley.Kei	thley2700
aux_ou	t_2 (pymeasure.instruments.srs.SR860 property),		method), 146	
	209	beep()	(pymeasure.instruments.keithley.Kei	thley6221
aux_ou	t_3 (pymeasure.instruments.srs.SR830 property),		method), 149	
	206	beep_st	ate (pymeasure.instruments.keithley.K	Ceithley2000
aux_ou	t_3 (pymeasure.instruments.srs.SR860 property),	haanan	property), 127	(2004)
2118 011	209 t_4 (pymeasure.instruments.srs.SR830 property),	peeber_	enabled sure.instruments.rohdeschwarz.sfm.SI	(pymea- EM
aux_ou	206		property), 187	11/1
aux_ou	t_4 (pymeasure.instruments.srs.SR860 property),	beeper_		(рутеа-
	209	•	sure.instruments.agilent.Agilent33220	• •
average	e_point (pymea-		property), 83	
	sure.instruments.anritsu.AnritsuMS9710C	BIAS (py	measure.instruments.agilent.agilentB1.	500.SamplingPostOutput
	property), 110	, .	attribute), 103	570
average	e_sweep (pymea-	b1as_en	tabled (pymeasure.instruments.srs.SR.	5/0 prop-
	sure.instruments.anritsu.AnritsuMS9710C property), 110	hias la	erty), 204 evel (pymeasure.instruments.srs.SR5)	70 nron-
averag	es (pymeasure.instruments.agilent.Agilent8722ES	DIU3_IC	erty), 204	το ριορ
	property), 69	binary_	values() (pymeasure.adapter	rs.Adapter
averag:	ing_enabled (pymea-	-	method), 31	•
	sure.instruments.agilent.Agilent8722ES	binary_	values() (pymeasure.adapters.Fai	keAdapter
	property), 69		method), 32	
axes (	pymeasure.instruments.newport.ESP300 prop-	binary_	values()	(pymea-
Avic (a	erty), 167 lass in pymeasure.instruments.attocube.anc300),		sure.adapters.PrologixAdapter 35	method),
AXIS (C	114	hinary	.values() (pymeasure.adapters.Seri	alAdanter
Axis (c	lass in pymeasure.instruments.newport.esp300),	Dillar y_	method), 33	an imprer
`	167	binary_	values() (pymeasure.adapters.Teln	etAdapter
AxisEr	ror (class in pymea-		method), 40	
	sure.instruments.newport.esp300), 168	binary_	values() (pymeasure.adapters.VIS	SAAdapter
В		, .	method), 37	1141 .
			values() (pymeasure.adapters.VXII	! I Adapter
BASE (py	measure.instruments.agilent.agilentB1500.Sampli	ingPostOut hinary	values()	(рутеа-
hacach	attribute), 103 eed (pymeasure.instruments.anaheimautomation.E			
basespi	property), 107	or seriesin	method), 164	1
basic_:	info(pymeasure.instruments.rohdeschwarz.sfm.SI	Γ <b>/β</b> KPreci	sion9130B (class in	рутеа-
	property), 187		sure.instruments.bkprecision), 115	
batch_:	size (pymeasure.instruments.pendulum.cnt91.CN	robµLank_£	front() (pymeasure.instruments	srs.SR570
1 ()	property), 184	blankin	method), 204  g (pymeasure.instruments.anapico.A	PSIN12G
beep()	(pymeasure.instruments.agilent.Agilent33220A method), 83	DIGIRII	property), 109	1511120
beep()	(pymeasure.instruments.agilent.Agilent33500	Boolean	Input (class in pymeasure.display.inp	uts), 50
occp()	method), 85		Parameter (class in	рутеа-
beep()	(pymeasure.instruments.agilent.Agilent34450A		sure.experiment.parameters), 44	
_	method), 73		(class in pymeasure.display.browser),	
beep()	(pymeasure.instruments.hp.HP33120A method),		Titem (class in pymeasure.display.brow	
	122		Widget (class in pymeasure.display.wi	_
beep()	(pymeasure.instruments.keithley.Keithley2000 method), 127	purier_	data (pymeasure.instruments.keithley. property), 127	хеншеу2000

246 Index

buffer_data(pymeasure.instruments.keithley.K	aithlan 210			
property), 135	eimiey240			,
buffer_data (pymeasure.instruments.keithley.K	oithlev245	calibra		(pymea-
property), 141	eiinie y 2 <del>4</del> 3	0	sure.instruments.rohdeschwarz.sfm.S.	FM
huffer data (nymeasure instruments keithley k	oithlev270	00	method), 188	
buffer_data (pymeasure.instruments.keithley.K property), 146	eiiniey270	'calibra		(pymea-
buffer_data(pymeasure.instruments.keithley.K	oithlov622	) 1		property),
property), 149	eiiniey022		124	
buffer_data(pymeasure.instruments.keithley.K	aithlan651	capacit	ance (pymeasure.instruments.agilent.a	Agilent34450A
property), 154	eiiniey051		property), 73	
buffer_frequency_time_series()	(рутеа-	capacit	ance_auto_range	(pymea-
sure.instruments.pendulum.cnt91.CNT			sure.instruments.agilent.Agilent34450	0A
method), 184	71		property), 73	
buffer_points	(nymaa	capacit	ance_range	(рутеа-
_	(pymea-		sure.instruments.agilent.Agilent34450	OA
sure.instruments.keithley.Keithley2000 erty), 127	prop-		property), 73	
buffer_points	(рутеа-	capacit	y (pymeasure.instruments.attocube.an	c300.Axis
sure.instruments.keithley.Keithley2400			property), 114	
	prop-	carrier	_enabled	(рутеа-
<pre>erty), 135 buffer_points</pre>	(nymaa		sure. instruments. rohdeschwarz. sfm. Set in the state of the state	ound_Channel
sure.instruments.keithley.Keithley2450	(pymea-		property), 196	
· · · · · · · · · · · · · · · · · · ·	prop-	carrier	_frequency	(рутеа-
erty), 141	(22222000		sure. instruments. rohdeschwarz. sfm. Set in the state of the state	ound_Channel
buffer_points	(pymea-		property), 196	
sure.instruments.keithley.Keithley2700	prop-	carrier		(рутеа-
erty), 146	(22222000		sure.instruments.rohdeschwarz.sfm. Set in the state of	ound_Channel
buffer_points	(pymea-		property), 196	
sure.instruments.keithley.Keithley6221	prop-	center_	at_peak()	(рутеа-
erty), 149	(22222000		sure.instruments.anritsu.AnritsuMS97	710C
buffer_points	(pymea-		method), 110	
sure.instruments.keithley.Keithley6517	D	center_	frequency	(рутеа-
<pre>property), 154 buffer_to_float()</pre>	(22222000		sure.instruments.advantest.advantest Particular (a) and the state of	R3767CG.AdvantestR3767C
	(pymea-		property), 66	
sure.instruments.signalrecovery.DSP72 method), 200	203	center_	frequency	(рутеа-
BufferCurve (class in pymeasure.display.curve	a) 40		sure.instruments.agilent.Agilent8257	D prop-
burst mode (nymeasure instruments agilent Agi	s), 49 Lont33220	Λ	erty), 67	
burst_mode (pymeasure.instruments.agilent.Agi property), 83	ieni332201	center_		(pymea-
burst_mode (pymeasure.instruments.agilent.Agi	Lant 2 2 5 0 0		sure.instruments.agilent.AgilentE440	8B
property), 86	<i>ieni33300</i>		property), 70	
burst_ncycles	(рутеа-	channel	(pymeasure.instruments.bkprecision.E	3KPrecision9130B
sure.instruments.agilent.Agilent332204			property), 115	
property), 83	1	channel	1 (pymeasure.instruments.srs.SR830	property),
burst_ncycles	(рутеа-		206	
sure.instruments.agilent.Agilent33500	prop-	channel.	2 (pymeasure.instruments.srs.SR830	property),
erty), 86	ргор-		206	
burst_period	(рутеа-	channel.	_down_relative()	(pymea-
sure.instruments.agilent.Agilent33500	prop-		sure.instruments.rohdeschwarz.sfm.S.	FM
erty), 86	ргор-		method), 188	
burst_state(pymeasure.instruments.agilent.Ag	ailant3322	channel.	_function	(рутеа-
property), 83	zuemisszzi	UA	sure.instruments.agilent.agilent4156.	SMU
burst_state(pymeasure.instruments.agilent.Ag	oilent3350	0	property), 79	
property), 86	5.11.11.13.30	channel.		(pymea-
busy (pymeasure.instruments.anaheimautomatio	n DPSorio	sMotorCo	sure.instruments.agilent.agilent4156.	VSU
property), 107				(
L L / / ,		channel.	_moue	(pymea-

	sure.instruments.agilent.agilent4156.SMU		(рутеа-
	property), 79	sure.instruments.keithley.Keithley2450	
channel	_mode (pymea-	method), 141	
	sure.instruments.agilent.agilent4156.VMU	<pre>check_errors()</pre>	(pymea-
	property), 82	sure.instruments.keithley.Keithley2600	
channel		method), 158	
	sure.instruments.agilent.agilent4156.VSU		(рутеа-
	property), 82	sure.instruments.keithley.Keithley2700	4.5
channel	_sweep_start (pymea-	method), 146	
	sure.instruments.rohdeschwarz.sfm.SFM		(рутеа-
	property), 188	sure.instruments.keithley.Keithley2750	47
channel	_sweep_step (pymea-	method), 157	
CHAINCE	sure.instruments.rohdeschwarz.sfm.SFM		(рутеа-
	property), 188	sure.instruments.keithley.Keithley6221	фуниси
channal	_sweep_stop (pymea-	method), 149	
CHAINCI	sure.instruments.rohdeschwarz.sfm.SFM		(рутеа-
	property), 188	sure.instruments.keithley.Keithley6517.	
channel		method), 154	Ь
Chamier	_table (pymea- sure.instruments.rohdeschwarz.sfm.SFM		(рутеа-
	· ·		
ah amma 1	property), 188	sure.instruments.keysight.KeysightDSC	X1102G
Chamiler	_up_relative() (pymea-		(
	sure.instruments.rohdeschwarz.sfm.SFM		(pymea-
, ,	method), 188	sure.instruments.keysight.KeysightN57	0/A
cnanneı	s_from_rows_columns() (pymea-		(
	sure.instruments.keithley.Keithley2700		
	method), 146	sure.display.widgets.EstimatorWidget	method),
check_a	cknowledgement() (pymea-	54	,
	sure.instruments.attocube.adapters.AttocubeCo		(pymea-
	method), 113	sure.instruments.agilent.agilentB1500.	AgilentB1500
check_e	* *	method), 94	
	sure.instruments.agilent.agilentB1500.AgilentB		(pymea-
	method), 94	sure.experiment.procedure.Procedure	method),
check_e			
	sure.instruments.agilent.agilentB1500.SMU	<pre>check_stop()</pre>	(pymea-
	method), 97	sure.display.windows.PlotterWindow	method),
check_e		59	
	sure.instruments.anaheimautomation.DPSeries	Mo <b>thrGoes</b> &H <b>ym</b> easure.experiment.parameters.List	Parameter
	method), 107	property), 45	
check_e	rrors() (pymea-	clear() (pymeasure.display.manager.Manager	method),
	sure.instruments.bkprecision.BKPrecision91301	3 53	
	method), 115	clear() (pymeasure.instruments.Instrument met	(hod), 61
check_e	rrors() (pymeasure.instruments.hp.HP3478A	clear() (pymeasure.instruments.keysight.Keysig	ghtDSOX1102G
	method), 124	method), 159	
check_e	rrors() (pymeasure.instruments.Instrument	<pre>clear_buffer()</pre>	(рутеа-
	method), 61	sure.instruments.agilent.agilentB1500	AgilentB1500
check_e	rrors() (pymea-	method), 94	
	sure.instruments.keithley.Keithley2000		(рутеа-
	method), 127	sure.instruments.agilent.Agilent33500 i	
check_e		86	//
	sure.instruments.keithley.Keithley2260B		(pymea-
	method), 133		method),
check_e		167	,
	Opymen		
	sure.instruments.keithley.Keithley2400	<pre>clear_overload() (pymeasure.instruments.sn</pre>	rs.SR570

clear_plot() (pymea-	61
sure.experiment.experiment.Experiment	complete (pymeasure.instruments.keithley.Keithley2000
method), 41	property), 127
clear_ramp_set() (pymea-	complete (pymeasure.instruments.keithley.Keithley2260B
sure.instruments.danfysik.Danfysik8500 method), 117	property), 133 complete (pymeasure.instruments.keithley.Keithley2400
clear_sequence() (pymea-	property), 135
sure.instruments.danfysik.Danfysik8500	complete (pymeasure.instruments.keithley.Keithley2450
method), 117	property), 141
clear_status() (pymea-	complete (pymeasure.instruments.keithley.Keithley2600
sure.instruments.keysight.KeysightDSOX1102G	property), 158
method), 159	complete (pymeasure.instruments.keithley.Keithley2700
clear_timer() (pymea-	property), 147
	1500mplete (pymeasure.instruments.keithley.Keithley2750
method), 94	property), 157
	complete (pymeasure.instruments.keithley.Keithley6221
method), 157	property), 149
close_rows_to_columns() (pymea-	
sure.instruments.keithley.Keithley2700 method), 146	property), 154
closed_channels (pymea-	complete (pymeasure.instruments.keysight.KeysightDSOX1102G property), 159
sure.instruments.keithley.Keithley2700 prop-	complete (pymeasure.instruments.keysight.KeysightN5767A
erty), 147	property), 163
closed_channels (pymea-	compliance (pymeasure.instruments.agilent.agilent4156.SMU
sure.instruments.keithley.Keithley2750 prop-	property), 80
erty), 157	compliance (pymeasure.instruments.agilent.agilent4156.VARD
CMU_MEASUREMENT (pymea-	property), 81
sure. instruments. a gilent. a gilent B1500. Wait Times the surface of the property of the p	Typnmpliance (pymeasure.instruments.agilent.agilent4156.VARX
attribute), 103	property), 81
CNT91 (class in pymeasure.instruments.pendulum.cnt91),	
184	sure.instruments.agilent.agilentB1500.MeasOpMode
coder_adjust() (pymea-	attribute), 102
sure.instruments.rohdeschwarz.sfm.SFM method), 188	compliance_current (pymea-
coder_id_frequency (pymea-	sure.instruments.keithley.Keithley2400 prop- erty), 135
sure.instruments.rohdeschwarz.sfm.SFM	compliance_current (pymea-
property), 188	sure.instruments.keithley.Keithley2450 prop-
coder_modulation_degree (pymea-	erty), 141
sure.instruments.rohdeschwarz.sfm.SFM	compliance_current (pymea-
property), 188	sure.instruments.yokogawa.Yokogawa7651
coder_pilot_deviation (pymea-	property), 215
sure. instruments. rohdeschwarz. sfm. SFM	COMPLIANCE_SIDE (pymea-
property), 189	sure. instruments. a gilent. a gilent B1500. Meas Op Mode
coder_pilot_frequency (pymea-	attribute), 102
sure.instruments.rohdeschwarz.sfm.SFM	compliance_voltage (pymea-
property), 189	sure.instruments.keithley.Keithley2400 prop-
coilconst (pymeasure.instruments.ami.AMI430 prop-	erty), 135
<pre>erty), 106 collapse_channel_string()</pre>	compliance_voltage (pymea-
collapse_channel_string() (pymea- sure.instruments.ni.virtualbench.VirtualBench	sure.instruments.keithley.Keithley2450 prop- erty), 141
method), 179	compliance_voltage (pymea-
complete (pymeasure.instruments.bkprecision.BKPrecision.	
property), 115	property), 215
complete (pymeasure.instruments.Instrument property),	

arms in atmum anta a ail ant a ail ant D 1500\ 102	configure consistence()
sure.instruments.agilent.agilentB1500), 103	
config_amplitude_modulation() (pymea-	
sure.instruments.agilent.Agilent8257D	method), 73
method), 67	configure_continuity() (pymea-
config_buffer() (pymea-	
sure.instruments.keithley.Keithley2000	method), 73
method), 127	configure_current() (pymea-
config_buffer() (pymea-	sure.instruments.agilent.Agilent34450A
sure.instruments.keithley.Keithley2400	method), 73
method), 135	configure_current_output() (pymea-
config_buffer() (pymea-	sure. instruments. ni. virtual bench. Virtual Bench. Power Supply
sure.instruments.keithley.Keithley2450	method), 178
method), 141	configure_dc_current() (pymea-
config_buffer() (pymea-	sure. instruments. ni. virtual bench. Virtual Bench. Digital Multimeter
sure.instruments.keithley.Keithley2700	method), 171
method), 147	configure_dc_voltage() (pymea-
config_buffer() (pymea-	
sure.instruments.keithley.Keithley6221	method), 171
method), 149	configure_diode() (pymea-
config_buffer() (pymea-	•
sure.instruments.keithley.Keithley6517B	method), 73
method), 154	configure_frequency() (pymea-
	- · · · · · · · · · · · · · · · · · · ·
config_low_freq_out() (pymea-	9 9
sure.instruments.agilent.Agilent8257D	method), 73
method), 67	configure_frequency_array_measurement()
config_pulse_modulation() (pymea-	
sure.instruments.agilent.Agilent8257D	method), 184
method), 67	configure_immediate_trigger() (pymea-
config_step_sweep() (pymea-	· ·
sure.instruments.agilent.Agilent8257D	method), 176
method), 67	configure_measurement() (pymea-
C:(C) (	6 A ail ant 1156 no in atmum anta ni vintu all an al Vintu all an al Diait al Multim at an
configure() (pymeasure.instruments.agilent.agilent415	6. A gilent 415 6 ure. instruments. ni. virtual bench. Virtual Bench. Digital Multimeter
configure() (pymeasure.instruments.aguent.aguent413 method), 78	o.Agueni41. <b>xu</b> re.instruments.nt.virtuatbencn.virtuatBench.Digitativiuttimeter method), 171
method), 78	
<pre>method), 78 configure_ac_current()</pre>	<pre>method), 171 configure_resistance()</pre>
method), 78 configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.I	method), 171 configure_resistance() (pymea- DigitalMultim <b>ute</b> rinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.I method), 171	method), 171 configure_resistance() (pymea- DigitalMultin <b>nate</b> rinstruments.agilent.Agilent34450A method), 74
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod)	method), 171  configure_resistance() (pymea- DigitalMultimuterinstruments.agilent.Agilent34450A  method), 74  configure_standard_waveform() (pymea-
method), 78  configure_ac_current() (pymea- sure.instruments.ni.virtualbench.VirtualBench.I method), 171  configure_analog_channel() (pymea- sure.instruments.ni.virtualbench.VirtualBench.II	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignalOsvėlinstrapaents.ni.virtualbench.VirtualBench.FunctionGenerator
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174	method), 171 configure_resistance() (pymea- DigitalMultin <b>nete</b> rinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignal@svėlinstrapnents.ni.virtualbench.VirtualBench.FunctionGenerator method), 173
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics()	method), 171 configure_resistance() (pymea- DigitalMultinnuterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignalOsrellinstrapments.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea-
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualBench	method), 171 configure_resistance() (pymea- DigitalMultinnarerinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignalOscélinstrapaents.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea- Bench.MixedSignalOscuiblenstxopgilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175	method), 171  configure_resistance() (pymea- DigitalMultimuterinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175	method), 171  configure_resistance() (pymea- DigitalMultimuterinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.Virtualbench.Virtualbench), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.Virtualbench	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignal@sreelinstrapments.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea- Bench.MixedSignid@sreillentstopgilent.Agilent34450A method), 74 configure_timing() (pymea- MixedSignal@sreelinstrapments.ni.virtualbench.VirtualBench.MixedSignalOscill
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Intervented), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Intervented), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.Virtualbench.Virtualbench), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.Virt	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignalOscillinstrapments.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea- Bench.MixedSignilOscillostropgilent.Agilent34450A method), 74 configure_timing() (pymea- MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBench.MixedSignalOscillostrapments.ni.virtualbench.VirtualBe
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.Virtualbench.Virtualbench), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.Virtualbench	method), 171 configure_resistance() (pymea- DigitalMultinnaterinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod).	method), 171  configure_resistance() (pymea- DigitalMultinumerinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175	method), 171  configure_resistance() (pymea- DigitalMultimuterinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_arbitrary_waveform() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175	method), 171 configure_resistance() (pymea- DigitalMultimuterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignalOscielinstrapments.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea- Bench.MixedSignidOsciillenstxxpgilent.Agilent34450A method), 74 configure_timing() (pymea- MixedSignalOscielinstrapments.ni.virtualbench.VirtualBench.MixedSignalOscill method), 176 configure_trigger_delay() (pymea- Bench.MixedSignidOsciillenstxxpievirtualbench.VirtualBench.MixedSignalOscill method), 176 configure_voltage() (pymea-
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_arbitrary_waveform() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod).	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignal@axelinstrapments.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea- Bench.MixedSignal@axillentsxxxpgilent.Agilent34450A method), 74 configure_timing() (pymea- MixedSignal@axelinstrapments.ni.virtualbench.VirtualBench.MixedSignalOscill method), 176 configure_trigger_delay() (pymea- Bench.MixedSignal@axillentxxxxpevirtualbench.VirtualBench.MixedSignalOscill method), 176 configure_voltage() (pymea- FunctionGenewatoinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.Virtualmethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.Virtualmethod), 175  configure_aralog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.Virtualmethod), 175  configure_arbitrary_waveform() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 172	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_aralog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_arbitrary_waveform() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 172  configure_arbitrary_waveform_gain_and_offset	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A method), 74 configure_standard_waveform() (pymea- MixedSignal@scellinstrapments.ni.virtualbench.VirtualBench.FunctionGenerator method), 173 configure_temperature() (pymea- Bench.MixedSignial@scillentscopgilent.Agilent34450A method), 74 configure_timing() (pymea- MixedSignal@scellinstrapments.ni.virtualbench.VirtualBench.MixedSignalOscillinethod), 176 configure_trigger_delay() (pymea- Bench.MixedSignal@scillentscopievirtualbench.VirtualBench.MixedSignalOscillinethod), 176 configure_voltage() (pymea- FunctionGenerateoinstruments.agilent.Agilent34450A method), 74 ()configure_voltage_output() (pymea-
method), 78  configure_ac_current() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 171  configure_analog_channel() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 174  configure_analog_channel_characteristics() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_analog_edge_trigger() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 175  configure_analog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_aralog_pulse_width_trigger() (pymeasure.instruments.ni.virtualbench.VirtualImethod), 175  configure_arbitrary_waveform() (pymeasure.instruments.ni.virtualbench.VirtualBench.Imethod), 172  configure_arbitrary_waveform_gain_and_offset	method), 171 configure_resistance() (pymea- DigitalMultimaterinstruments.agilent.Agilent34450A

constant_value	(pymea-		47	
sure.instruments.agilent.agilent4156.S	SMU	current	(pymeasure.instruments.agilent.Agilen	t34450A
property), 80			property), 74	
constant_value	(рутеа-	CURRENT	(pymeasure.instruments.agilent.agilent	B1500.MeasOpMode
sure.instruments.agilent.agilent4156.V	/SU		attribute), 102	
property), 82		current	(pymeasure.instruments.bkprecision.Bk	(Precision9130B
contact_current_1	(рутеа-		property), 115	
sure.instruments.razorbill.razorbillRP		current	(pymeasure.instruments.danfysik.Danf	vsik8500
property), 185			property), 117	,
contact_current_2	(рутеа-	current	(pymeasure.instruments.deltaelektronik	a.SM7045D
sure.instruments.razorbill.razorbillRP	4 .		property), 119	
property), 185	100	current	(pymeasure.instruments.keithley.Keith	alev2000
contact_voltage_1	(рутеа-	current	property), 127	iic)2000
sure.instruments.razorbill.razorbillRP		current	(pymeasure.instruments.keithley.Keithl	ev2260R
property), 185	100	Cullenc	property), 133	еу2200В
	(mmaga	cummon+		Jan 2400
contact_voltage_2	(pymea-	current	(pymeasure.instruments.keithley.Keith	uey2400
sure.instruments.razorbill.razorbillRP	100		property), 135	1 2450
property), 185	,	current	(pymeasure.instruments.keithley.Keith	ıley2450
<pre>continue_single_sweep()</pre>	(рутеа-		property), 141	65.15D
sure.instruments.rohdeschwarz.fsl.FSI	_	current	(pymeasure.instruments.keithley.Keithl	ey6517B
method), 199			property), 154	
${\tt continuity} \ (py measure. instruments. a gilent. A g$	ilent34450.	<i>A</i> current		ghtN5767A
property), 74			property), 163	
continuous (pymeasure.instruments.pendulum.	.cnt91.CNT	<i>Scl</i> urrent	_ac(pymeasure.instruments.agilent.Agi	lent34410A
property), 184			property), 72	
continuous_sweep	(pymea-	current	_ac (pymeasure.instruments.agilent.Agi	lent34450A
sure.instruments.rohdeschwarz.fsl.FSL	_ prop-		property), 74	
erty), 199		current	_ac (pymeasure.instruments.hp.Hl	P34401A
<pre>control() (pymeasure.instruments.Instruments.</pre>	nt static		property), 123	
method), 61		current	_ac_auto_range	(pymea-
control() (pymeasure.instruments.keysight.Ke	ysightDSO.			1
static method), 159	. 0		property), 74	
control_mode	(pvmea-	current	_ac_bandwidth	(pymea-
sure.instruments.oxfordinstruments.IT			sure.instruments.keithley.Keithley2000	4.5
property), 181	0000		<i>erty</i> ), 127	P.op
controllerBoardVersion	(nymea-	current	_ac_digits	(рутеа-
sure.instruments.attocube.anc300.ANC			sure.instruments.keithley.Keithley2000	
	Soconin	ouci	erty), 127	ргор
property), 113	(mma a a	cummon+		(mym a a
convert_timestamp_to_values()	(pymea-	current	_ac_nplc	(рутеа-
sure.instruments.ni.virtualbench.Virtu	ш <b>ь</b> енсн		sure.instruments.keithley.Keithley2000	prop-
method), 179	,		erty), 127	/
<pre>convert_values_to_datetime()</pre>	(рутеа-	current	_ac_range	(pymea-
sure.instruments.ni.virtualbench.Virtu	alBench		sure.instruments.agilent.Agilent34450A	l
method), 179			property), 74	
<pre>convert_values_to_timestamp()</pre>	(рутеа-	current	_ac_range	(pymea-
sure.instruments.ni.virtualbench.Virtu	alBench		sure.instruments.keithley.Keithley2000	prop-
method), 179			erty), 127	
<pre>create_filename() (in module</pre>	рутеа-	current	_ac_reference	(pymea-
sure.experiment.experiment), 42			sure.instruments.keithley.Keithley2000	prop-
<pre>create_marker()</pre>	(рутеа-		erty), 128	
sure.instruments.rohdeschwarz.fsl.FSI		current	_ac_resolution	(рутеа-
method), 199			sure.instruments.agilent.Agilent34450A	= :
Crosshairs (class in pymeasure.display.curves	s), 49		property), 74	
			**:	

	sure.instruments.agilent.Agilent34450£ property), 74	4		sure.instruments.keithley.Keithley2450 erty), 142	prop-
	_dc (pymeasure.instruments.agilent.Agi	ilent34410	Acurr	* *	рутеа-
	property), 72			sure.instruments.keithley.Keithley6517B	
current_		P34401A		property), 154	
	property), 123		curr	ent_range (	рутеа-
current_	_digits	(pymea-		sure.instruments.keysight.KeysightN576	7A
	sure.instruments.keithley.Keithley2000	prop-		property), 163	
	erty), 128		curr	ent_reference ()	рутеа-
current_	_filter_count	(pymea-		sure.instruments.keithley.Keithley2000	prop-
	sure.instruments.keithley.Keithley2450	prop-		erty), 128	
	erty), 141		curr	ent_resolution ()	рутеа-
current_	_filter_state	(рутеа-		sure.instruments.agilent.Agilent34450A	
	sure.instruments.keithley.Keithley 2450	prop-		property), 75	
	erty), 141		curr	-	рутеа-
		(рутеа-		sure.instruments.danfysik.Danfysik8500	
	sure.instruments.keithley.Keithley2450	prop-		property), 117	
	erty), 141		curv	•	рутеа-
current_		(рутеа-		sure.instruments.signalrecovery.DSP726	55
	sure.instruments.keithley.Keithley2260	B		property), 201	
	property), 133		curv	•	рутеа-
current_		(рутеа-		sure.instruments.signalrecovery.DSP726	55
	sure.instruments.yokogawa.Yokogawa C	GS200		property), 201	
	property), 216		curv	-	рутеа-
current_		(рутеа-		sure.instruments.signalrecovery.DSP726	55
	sure.instruments.agilent.agilent4156.Si	MU		property), 201	
	property), 80		curv	•	рутеа-
current_	_	(рутеа-		sure.instruments.signalrecovery.DSP726	55
	sure.instruments.keithley.Keithley2000	prop-		property), 201	
	erty), 128		Cust	`	рутеа-
current_	_	(рутеа-	_	sure.instruments.agilent.agilentB1500),	
	sure.instruments.keithley.Keithley2400	prop-	CW_f		рутеа-
	erty), 135	,		sure.instruments.rohdeschwarz.sfm.SFM	
current_		(рутеа-		property), 189	
	sure.instruments.keithley.Keithley2450 erty), 142		D		
current_		(рутеа-	dac1	(pymeasure.instruments.ametek.Ametek7270	) prop-
	sure.instruments.keithley.Keithley6517	В		erty), 104	
	property), 154	,	dac1	(pymeasure.instruments.signalrecovery.DS	SP7265
	_output_off_state	(рутеа-		property), 201	
	sure.instruments.keithley.Keithley2450	prop-		$(pymeasure.instruments.srs.SR830\ property)$	
	erty), 142		dac1	(pymeasure.instruments.srs.SR860 property)	, 209
current_	_ppm(pymeasure.instruments.danfysik.1 property), 117	Danfysik85	Wac2	(pymeasure.instruments.ametek.Ametek7270 erty), 104	) prop-
current_	range	(рутеа-	dac2	(pymeasure.instruments.signalrecovery.DS	SP7265
	sure.instruments.agilent.Agilent34450A	4		property), 201	
	property), 74		dac2	(pymeasure.instruments.srs.SR830 property)	, 206
current_	range	(pymea-		(pymeasure.instruments.srs.SR860 property)	
	sure.instruments.keithley.Keithley2000	prop-		(pymeasure.instruments.ametek.Ametek7270	
	erty), 128			erty), 104	1 · F
current_	_	(pymea-	dac3	(pymeasure.instruments.signalrecovery.DS	SP7265
	sure.instruments.keithley.Keithley 2400	prop-		property), 201	
	erty), 135		dac3	(pymeasure.instruments.srs.SR830 property)	, 206
current_	range	(рутеа-		(pymeasure.instruments.srs.SR860 property)	

dac4 (pymeasure.instruments.ametek.Ametek72 erty), 104		data_vo	sure. instruments. a gilent. A gilent 3.	(pymea- 3500 method),
dac4 (pymeasure.instruments.signalrecovery.liproperty), 201	DSP7265	date	86 (pymeasure.instruments.rohdesch	warz.sfm.SFM
dac4 (pymeasure.instruments.srs.SR830 propert	y), 206		property), 189	
dac4 (pymeasure.instruments.srs.SR860 propert	y), 209	dc_mode	() (pymeasure.instruments.lakesh	ore.LakeShore425
Danfysik8500 (class in	рутеа-		method), 166	
sure.instruments.danfysik), 116		dcmode	(pymeasure.instruments.srs.SR86	60 property),
DanfysikAdapter (class in	рутеа-		209	
sure.instruments.danfysik), 116		decode_	mode() (pymeasure.instruments	s.hp.HP3478A
data (pymeasure.experiment.experiment.Ex	periment		class method), 124	
property), 41	_	decode_	range() (pymeasure.instruments	s.hp.HP3478A
data (pymeasure.instruments.agilent.Agiler	ıt8722ES		class method), 124	•
property), 69		decode_	status()	(рутеа-
data_arb() (pymeasure.instruments.agilent.Ag	ilent33500		sure.instruments.hp.HP3478A cl	lass method),
method), 86			124	,,
data_complex	(pymea-	decode_	trigger()	(pymea-
sure.instruments.agilent.Agilent8722E			sure.instruments.hp.HP3478A st	
property), 69			125	·····,
data_format()	(pymea-	default	_setup()	(pymea-
sure.instruments.agilent.agilentB1500.			sure.instruments.keysight.Keysigh	* *
method), 94			method), 160	
data_log_magnitude	(nymea-	define	arbitary_waveform()	(рутеа-
sure.instruments.agilent.Agilent8722E		uciinc_	sure.instruments.keithley.Keithley	
property), 69	5		method), 149	.0221
data_magnitude	(nvmea-	define	position()	(рутеа-
sure.instruments.agilent.Agilent8722E		uciinc_	sure.instruments.newport.esp300.	
property), 69	5		167	and memod),
data_memory_a_condition	(рутеа-	delav t	ime (pymeasure.instruments.agiler	nt agilent4156 Agilent4156
sure.instruments.anritsu.AnritsuMS97.		ueray_t	property), 78	m.ugnem+130.11gnem+130
property), 110	100	dotocto	dfrequency (pymeasure.instrume	ente ere CR860
data_memory_a_size	(рутеа-	uetecte	property), 209	ents.srs.srcoo
sure.instruments.anritsu.AnritsuMS97.		dotormi	ne_valid_channels()	(nym a a
	100	ue cermi	sure.instruments.keithley.Keithley	(pymea- ,2700
property), 110	(222220000			/2/00
data_memory_a_values sure.instruments.anritsu.AnritsuMS97.	(pymea-	doviati	method), 147	schwarz stm Sound Channal
	100	ueviati	on (pymeasure.instruments.rohdes	chwarz.sjm.souna_channet
property), 110	(222220000	diai+in	property), 196 e() (pymeasure.instruments.keysig	aht VansiahtDSOV1102C
data_memory_b_condition sure.instruments.anritsu.AnritsuMS97	(pymea-	urgitiz	method), 160	gm.KeysigmDSOX1102G
	10C	diode		1 ail ant 24.4504
property), 111	(	aroue	(pymeasure.instruments.agilent.A	Agueni34430A
data_memory_b_size	(pymea-	din coo	property), 75	a. A. mita. MS0710C
sure.instruments.anritsu.AnritsuMS97	10C	aip_sea	rch (pymeasure.instruments.anrits	su.AnrusuMS9/10C
property), 111	(	44	property), 111	in and an adian DDC anian Madan C
data_memory_b_values	(pymea-	arrecti	on (pymeasure.instruments.anahei	maniomanion.DFSeriesMotorCo
sure.instruments.anritsu.AnritsuMS97	10C	D:	property), 107	
property), 111	(	Directo	ryLineEdit (class in	рутеа-
data_memory_select	(pymea-	dicabla	sure.display.widgets), 54	ailant 1156 CMII
sure.instruments.anritsu.AnritsuMS97	10C	aisabie	(pymeasure.instruments.agilent.ag	guent4130.SMU
property), 111	.1 497335	(C) 1- 1 -	property), 80	
data_phase (pymeasure.instruments.agilent.Ag	uento/22E	misabie	= :	gueni4130.VMU
property), 69	(mayor : : :	dicabl-	property), 82	ailant4156 VCII
data_variables	(pymea-		(pymeasure.instruments.agilent.ag	guem4130.V3U
sure.instruments.agilent.agilent4156.A	iguent4136		property), 82	4 Apilout0257D
property), 78		ursabre	() (pymeasure.instruments.agilent	.Aguenio23/ <b>D</b>

	method), 67			method), 67	
disable	() (pymeasure.instruments.agilent.agile	entB1500.S	<i>NHI</i> sable	_modulation()	(pymea-
	method), 97			sure.instruments.agilent.Agilent8257D	
disable	() (pymeasure.instruments.anritsu.Anri	tsuMG369.	2 <i>C</i>	method), 67	
	method), 110		disable	_offset_current()	(pymea-
disable	() (pymeasure.instruments.danfysik.Da	nfysik8500		sure.instruments.srs.SR570 method), 2	04
	method), 117		disable	_output_trigger()	(pymea-
disable	() (pymeasure.instruments.deltaelektro	nika.SM70	45D	sure. instruments. keithley. Keithley 2400	
	method), 119			<i>method</i> ), 135	
disable	() (pymeasure.instruments.keysight.Key	vsightN576	724isable	_output_trigger()	(pymea-
	method), 163			sure. instruments. keithley. Keithley 6221	
disable	() (pymeasure.instruments.newport			method), 150	
	method), 167		disable	_persistent_switch()	(pymea-
	() (pymeasure.instruments.newport.esp			sure.instruments.ami.AMI430 method).	
	method), 167		disable	_pulse_modulation()	(рутеа-
	() (pymeasure.instruments.parker.Pa	rkerGV6		sure.instruments.agilent.Agilent8257D	
	method), 183			method), 67	
disable_		* *		_reference()	(рутеа-
	sure.instruments.agilent.agilent4156.A	gilent4156		sure.instruments.keithley.Keithley2000	
	method), 78			method), 128	
	_amplitude_modulation()		disable.		(pymea-
	sure.instruments.agilent.Agilent8257D			sure.instruments.anapico.APSIN12G	method),
	method), 67	,		109	,
	_averaging()		disable.	_source()	(рутеа-
	sure.instruments.agilent.Agilent8722E	2		sure.instruments.keithley.Keithley2400	
	method), 69	CD 570	421.1 -	method), 135	(
disable_	* * * * * * * * * * * * * * * * * * * *	rs.SK5/U	alsable.		(рутеа-
	method), 204	(		sure.instruments.keithley.Keithley2450	
	_buffer()	(рутеа-	dicable	method), 142	(
	sure.instruments.keithley.Keithley2000		aisabie.	_source()	(рутеа-
	method), 128	(		sure.instruments.keithley.Keithley6221	
	_buffer() sure.instruments.keithley.Keithley2400	(pymea-	dicable	method), 150 _source()	(122222000
	method), 135		uisabie.	_source() sure.instruments.keithley.Keithley6517.	(pymea-
	_buffer()	(nymaa		method), 154	D
	sure.instruments.keithley.Keithley2450	(рутеа-	disable	_source()	(рутеа-
	method), 142		uisubic.	_sure.instruments.yokogawa.Yokogawa7	
		(рутеа-		method), 215	051
	sure.instruments.keithley.Keithley2700		display		nt33500
	method), 147		uropruj	property), 86	
	_buffer()	(pymea-	display	_closed_channels()	(рутеа-
	sure.instruments.keithley.Keithley6221	футен	uropray.	sure.instruments.keithley.Keithley2700	футеа
	method), 150			method), 147	
	_buffer()	(рутеа-	display	_enabled	(рутеа-
	sure.instruments.keithley.Keithley6517			sure.instruments.keithley.Keithley2400	
	method), 154			erty), 135	r
	_filter()	(рутеа-	display	_enabled	(pymea-
	sure.instruments.keithley.Keithley2000		_ ,	sure.instruments.keithley.Keithley6221	prop-
	method), 128			erty), 150	
	_heater()	(рутеа-	display	_estimates()	(pymea-
	sure.instruments.lakeshore.LakeShore3			sure.display.widgets.EstimatorWidget	
	method), 165			54	
disable_	_low_freq_out()	(pymea-	display.	_reset()	(pymea-
	sure.instruments.agilent.Agilent8257D			sure.instruments.hp.HP3478A method)	, 125

display_text (pymeasure.instruments.hp.HP3478A property), 125	sure.instruments.keithley.Keithley2000	(pymea-
display_text (pymea- sure.instruments.keithley.Keithley2700 prop- erty), 147	<pre>method), 128 enable_low_freq_out() sure.instruments.agilent.Agilent8257D</pre>	(pymea-
display_text_no_symbol (pymea- sure.instruments.hp.HP3478A property), 125	<pre>method), 67 enable_offset_current()     sure.instruments.srs.SR570 method), 2</pre>	(рутеа-
download_data() (pymea- sure.instruments.keysight.KeysightDSOX1102G	<pre>enable_persistent_switch()     sure.instruments.ami.AMI430 method)</pre>	( <i>pymea</i> - , 106
method), 160 download_image() (pymeasure.instruments.keysight.KeysightDSOX1102G	enable_pulse_modulation() sure.instruments.agilent.Agilent8257D method), 67	(pymea-
method), 160  DPSeriesMotorController (class in pymeasure.instruments.anaheimautomation), 107	enable_reference() sure.instruments.keithley.Keithley2000 method), 128	(pymea-
DSP7265 (class in pymea- sure.instruments.signalrecovery), 200	enable_rf() (pymeasure.instruments.anapico.amethod), 109	
dwell_time (pymeasure.instruments.agilent.Agilent82571 property), 67	Oenable_source() sure.instruments.keithley.Keithley2400 method), 135	(pymea-
E echo() (pymeasure.instruments.parker.ParkerGV6	enable_source() sure.instruments.keithley.Keithley2450	(pymea-
method), 183 emit() (pymeasure.display.log.LogHandler method), 52 emit() (pymeasure.experiment.workers.Worker method),	method), 142 enable_source() sure.instruments.keithley.Keithley6221	(рутеа-
46 enable() (pymeasure.instruments.agilent.Agilent8257D	<pre>method), 150 enable_source()</pre>	(рутеа-
method), 67 enable() (pymeasure.instruments.agilent.agilentB1500.Samethod), 97	sure.instruments.keithley.Keithley6517 MU method), 155 enable_source()	(рутеа-
enable() (pymeasure.instruments.anritsu.AnritsuMG3692 method). 110	method), 215	
enable() (pymeasure.instruments.danfysik.Danfysik8500 method), 117 enable() (pymeasure.instruments.deltaelektronika.SM704	property), 133	
method), 119 enable() (pymeasure.instruments.keysight.KeysightN576)	property), 168	(pymea-
method), 163 enable() (pymeasure.instruments.newport.ESP300 method), 167	<pre>property), 108 encoder_delay</pre>	(pymea-
enable() (pymeasure.instruments.newport.esp300.Axis method), 168	sure.instruments.anaheimautomation.l property), 108 encoder_enabled	DPSeriesMotorController (pymea-
enable() (pymeasure.instruments.parker.ParkerGV6 method), 183 enable_amplitude_modulation() (pymea-	sure.instruments.anaheimautomation.l property), 108	
sure.instruments.agilent.Agilent8257D method), 67	encoder_motor_ratio sure.instruments.anaheimautomation.l	(pymea- DPSeriesMotorController
enable_averaging() (pymea- sure.instruments.agilent.Agilent8722ES method), 69	property), 108 encoder_retries sure.instruments.anaheimautomation.1	(pymea- OPSeriesMotorController
enable_bias() (pymeasure.instruments.srs.SR570 method), 204	property), 108 encoder_window sure.instruments.anaheimautomation.1	(pymea- DPSeriesMotorController

property), 214 property), 189 err_status (pymeasure.instruments.srs.SR830 prop- ext_ref_extension (pymea-erty), 206 sure.instruments.rohdeschwarz.sfm.SFM	
error (pymeasure.instruments.keithley.Keithley2260B property), 189 property), 133 ext_trig_out (pymea-	
error (pymeasure.instruments.keithley.Keithley2400 sure.instruments.agilent.Agilent33500 propproperty), 135 erty), 86	
error (pymeasure.instruments.keithley.Keithley2450 ext_vid_connector (pymea-property), 142 sure.instruments.rohdeschwarz.sfm.SFM	
error (pymeasure.instruments.keithley.Keithley2600 property), 189 property), 158 external_arming_start_slope (pymea-	
error (pymeasure.instruments.keithley.Keithley2700 sure.instruments.pendulum.cnt91.CNT91 property), 147 property), 184	
error (pymeasure.instruments.keithley.Keithley6221 external_modulation_frequency (pymea-property), 150 sure.instruments.rohdeschwarz.sfm.SFM	
error (pymeasure.instruments.keithley.Keithley6517B property), 189 property), 155 external_modulation_power (pymea-	
error (pymeasure.instruments.newport.ESP300 prop- erty), 167 sure.instruments.rohdeschwarz.sfm.SFM property), 189	
error_reg (pymeasure.instruments.anaheimautomation.D <b>PseriexsMa</b> dto <b>m6dn:lartiro</b> n_source (pymea-property), 108 sure.instruments.rohdeschwarz.sfm.SFM	
error_status (pymeasure.instruments.hp.HP3478A property), 190 property), 125 external_start_arming_source (pymea-	
errors (pymeasure.instruments.newport.ESP300 prop- erty), 167 sure.instruments.pendulum.cnt91.CNT91 property), 184	
ese2 (pymeasure.instruments.anritsu.AnritsuMS9710C extfreqency (pymeasure.instruments.srs.SR860 prop- property), 111 erty), 209	
ESP300 (class in pymeasure.instruments.newport), 167 esr2 (pymeasure.instruments.anritsu.AnritsuMS9710C property), 111 extract_value() (pymeasure.instruments.attocube.adapters.AttocubeConsoleAdapter	ter
EstimatorThread (class in pymeasure.display.widgets), 54	
EstimatorWidget (class in pymeasure.display.widgets), factory_reset() (pymeasure.instruments.keysight.KeysightDSOX1102G	
eval_string() (pymea-method), 160 sure.display.widgets.SequencerWidget static FakeAdapter (class in pymeasure.adapters), 32	
method), 56 field (pymeasure.instruments.ami.AMI430 property), event_reg(pymeasure.instruments.rohdeschwarz.sfm.SFM 106	
property), 189  execute() (pymeasure.experiment.procedure.Procedure  method), 43  field (pymeasure.instruments.fwbell.FWBell5080 property), 121  field (pymeasure.instruments.lakeshore.LakeShore425)	
method), 43 expand_channel_string() (pymea- sure.instruments.ni.virtualbench.VirtualBench method), 180  field (pymeasure.instruments.lakeshore.LakeShore425 property), 166 fields() (pymeasure.instruments.fwbell.FWBell5080 method), 121	
Experiment (class in pymeasure.display.manager), 52 Experiment (class in pymeasure.display.manager), 52 Experiment (class in pymeasure.display.manager), 52 property), 209	
sure.experiment.experiment), 41 filter (pymeasure.instruments.agilent.agilentB1500.SMU  ExperimentQueue (class in pymea-property), 97  pymea-property), 97	
sure.display.manager), 52 filter (pymeasure.instruments.ni.virtualbench.VirtualBench.Funct.export_signal() (pymeasure.instruments.ni.virtualbench.VirtualBench.Distrullented (pymeasure.instruments.srs.SR860 method), 170 property), 209	ionGen

filter_count	(pymea-	property), 83	
sure.instruments.keithley.Keithley24	100 prop-	frequency (pymeasure.instruments.agilent.Agile	nt33500
erty), 135		property), 86	
filter_slope (pymeasure.instruments.srs.S	R830 prop-	${\tt frequency}(py measure. instruments. a gilent. A gilenter a g$	nt33521A
erty), 206		property), 88	
filter_slope (pymeasure.instruments.srs.S	R860 prop-	${\tt frequency} \ ({\it pymeasure.instruments.agilent.Agile}$	nt34450A
erty), 209		property), 75	
filter_state	(pymea-	${\tt frequency} \ ({\it pymeasure.instruments.agilent.Agile}$	nt8257D
sure.instruments.keithley.Keithley24	100 prop-	property), 68	T 4000
erty), 136	77 1.11 0.41	frequency (pymeasure.instruments.agilent.Agile	ntE4980
filter_type (pymeasure.instruments.keithle	y.Keithley240		17070
property), 136	D570	frequency (pymeasure.instruments.ametek.Ame	tek/2/0
filter_type (pymeasure.instruments.srs.S.	K5/U prop-	property), 104	CINIAC
erty), 204	(	frequency (pymeasure.instruments.anapico.AP	SIN12G
find_img_index()	(pymea-	property), 109	MC2602C
sure.display.curves.ResultsImage 50	method),	frequency (pymeasure.instruments.anritsu.Anrit	Sulvi G3092C
	uta) 50	property), 110	200 Aria
FloatInput ( <i>class in pymeasure.display.inp</i> FloatParameter ( <i>class in</i>		frequency (pymeasure.instruments.attocube.anc property), 114	300.Axis
sure.experiment.parameters), 44	рутеа-	frequency (pymeasure.instruments.hp.HP33120	M nron
Fluke7341 (class in pymeasure.instruments.)	Auke) 120	erty), 122	n prop-
		(Ufrequency (pymeasure.instruments.keithley.Keith	hlev2000
method), 98	cmB1500.5M	property), 128	ucy2000
	t avilentR150	ргорску), 120 O <b>EA edwealDD (O</b> ymeasure.instruments.rohdeschwar	7 sfm SFM
method), 94	i.ugiieiiiB150	property), 190	2.5/11.51 1/1
	t.agilentB150	OE <b>SAM</b> iency (pymeasure.instruments.rohdeschwar	z.sfm.Sound Channel
method), 97		property), 196	
FORCE_SIDE (pymeasure.instruments.agilent.	agilentB1500	.Meeg@pMeydpymeasure.instruments.signalrecove	ry.DSP7265
attribute), 102		property), 201	
force_trigger()	(pymea-	frequency (pymeasure.instruments.srs.SR510 pr	operty),
sure.instruments.ni.virtualbench.Vii	rtualBench.M	ixedSignal <b>Q\d</b> illoscope	
method), 176		frequency (pymeasure.instruments.srs.SR830 pr	operty),
format (pymeasure.instruments.pendulum.ca	nt91.CNT91	206	
property), 184		frequency (pymeasure.instruments.srs.SR860 pr	operty),
<pre>format() (pymeasure.experiment.results.CS</pre>	VFormatter	209	
method), 47			(рутеа-
format() (pymeasure.experiment.res	sults.Results	sure.instruments.keithley.Keithley2000	prop-
method), 47		erty), 128	
<pre>freq_center(pymeasure.instruments.rohdes</pre>	schwarz.fsl.FS		(рутеа-
property), 199	41.561	sure.instruments.agilent.Agilent34450A	
freq_span(pymeasure.instruments.rohdesch	warz.fsl.FSL	property), 75	,
property), 199	. C.LEG	- · ·	(рутеа-
freq_start(pymeasure.instruments.rohdesc	rhwarz.fsl.FSI		
property), 199	f-1 FCI	property), 75	(
freq_stop(pymeasure.instruments.rohdesch	warz.jsi.FSL		(рутеа-
property), 199	(mmag	sure.instruments.agilent.Agilent34450A	
freq_sweep()	(pymea-	<pre>property), 75 frequency_digits</pre>	(nymag
sure.instruments.agilent.AgilentE49 method), 71	.00	sure.instruments.keithley.Keithley2000	(pymea- prop-
frequencies (pymeasure.instruments.agilen	t Agilent8777		prop-
property), 69	15 1101110122		(рутеа-
frequencies (pymeasure.instruments.agilen	t AgilentF440		
property), 70		property), 190	=
frequency (nymeasure instruments agilent A	oilent33220A		(nymea-

sure.instruments.agilent.AgilentE4408B	method), 201
property), 70	get_buffer() (pymeasure.instruments.srs.SR830
frequency_reference (pymea- sure.instruments.keithley.Keithley2000 prop-	<pre>method), 207 get_calibration_information()</pre>
erty), 128	<pre>get_calibration_information()</pre>
	method), 180
sure.instruments.agilent.AgilentE4408B	<pre>get_data() (pymeasure.instruments.agilent.agilent4156.Agilent4156</pre>
<pre>property), 70 frequency_threshold (pymea-</pre>	
	get_estimates() (pymea-
sure.instruments.keithley.Keithley2000 property), 129	sure.display.widgets.EstimatorWidget method), 54
	<pre>get_estimates()</pre>
sure.instruments.agilent.Agilent34450A	sure.experiment.procedure.Procedure method),
property), 75	43
	<pre>get_library_version()</pre>
sure.instruments.agilent.Agilent34450A property), 75	sure.instruments.ni.virtualbench.VirtualBench method), 180
<pre>frequencypreset1 (pymeasure.instruments.srs.SR860</pre>	<pre>get_noise_bandwidth (pymea-</pre>
property), 209	sure.instruments.srs.SR860 property), 210
frequencypreset2 (pymeasure.instruments.srs.SR860	<pre>get_procedure()</pre>
property), 210	sure.display.widgets.InputsWidget method),
frequencypreset3 (pymeasure.instruments.srs.SR860	55
property), 210	<pre>get_scaling() (pymeasure.instruments.srs.SR830</pre>
frequencypreset4 (pymeasure.instruments.srs.SR860	method), 207
property), 210	<pre>get_sequence_from_tree()</pre>
front_blanked (pymeasure.instruments.srs.SR570	sure.display.widgets.SequencerWidget method),
property), 204	56
<pre>front_panel (pymeasure.instruments.srs.SR860 prop-</pre>	<pre>get_signal_strength_indicator (pymea-</pre>
erty), 210	sure.instruments.srs.SR860 property), 210
FSL (class in pymeasure.instruments.rohdeschwarz.fsl),	<pre>get_state_of_channels()</pre>
199	sure.instruments.keithley.Keithley2700
FWBell5080 (class in pymeasure.instruments.fwbell),	method), 147
121	<pre>get_status() (pymeasure.instruments.hp.HP3478A</pre>
G	<pre>get_time() (pymeasure.instruments.Mock method), 63</pre>
gain_mode (pymeasure.instruments.srs.SR570 property), 204	<pre>get_voltage() (pymeasure.instruments.Mock method), 63</pre>
gasflow (pymeasure.instruments.oxfordinstruments.ITC50	
property), 181	getAI() (in module pymeasure.instruments.comedi), 65
gen_measurement() (pymea-	getAO() (in module pymeasure.instruments.comedi), 65
sure.experiment.procedure.Procedure method), 43	gettimebase (pymeasure.instruments.srs.SR860 property), 210
GeneralError (class in pymea- sure.instruments.newport.esp300), 168	gpib() (pymeasure.adapters.PrologixAdapter method), 35
get() (pymeasure.instruments.agilent.agilentB1500.Custo	on <b>gaib</b> naddress (pymea-
class method), 101	sure.instruments.rohdeschwarz.sfm.SFM
<pre>get_array() (in module pymea-</pre>	property), 190
sure.experiment.experiment), 42	GPIB_trigger() (pymeasure.instruments.hp.HP3478A
get_array_steps() (in module pymea-	method), 123
sure.experiment.experiment), 42	ground_all() (pymea-
	2 · · · - · - · - · · - · · · · · · · ·
get array zero() (in module nymea-	sure.instruments.attocube.anc300.ANC300Controller
get_array_zero() (in module pymea- sure.experiment.experiment), 42	sure.instruments.attocube.anc300.ANC300Controller method), 113
<pre>get_array_zero() (in module pymea- sure.experiment.experiment), 42 get_buffer() (pymea-</pre>	sure.instruments.attocube.anc300.ANC300Controller method), 113

H		ho	rizontal_time_div (pymea-
handle_abort()	(рутеа-		sure.instruments.srs.SR860 property), 210
sure.experiment.workers.Worker	method),		33120A (class in pymeasure.instruments.hp), 122
46	,,,		34401A (class in pymeasure.instruments.hp), 123
handle_error()	(рутеа-		3478A (class in pymeasure.instruments.hp), 123
sure.experiment.workers.Worker	method),	HP:	3478A.ERRORS (class in pymeasure.instruments.hp),
46			123
harmonic (pymeasure.instruments.ametek.A property), 104	metek7270		3478A. SRQ (class in pymeasure.instruments.hp), 123 ADC (pymeasure.instruments.agilent.agilentB1500.ADCType
harmonic (pymeasure.instruments.signalrecov property), 202	ery.DSP726	55 HS <i>I</i>	attribute), 101 ADC (pymeasure.instruments.agilent.agilentB1500.ADCType
harmonic (pymeasure.instruments.srs.SR830	property),	HSA	attribute), 101 ADC_PULSED (pymea-
harmonic (pymeasure.instruments.srs.SR860	nronerty)		sure.instruments.agilent.agilentB1500.ADCType
210			attribute), 101
harmonicdual (pymeasure.instruments.srs.SF erty), 210		I	
has_amplitude_modulation	(рутеа-	id	(pymeasure.instruments.advantest.advantestR3767CG.AdvantestR3767C
sure.instruments.agilent.Agilent8257	D prop-		property), 66
<i>erty</i> ), 68	(	10	(pymeasure.instruments.agilent.Agilent33500 prop-
has_modulation	(pymea-	- 4	erty), 86
sure.instruments.agilent.Agilent8257	D prop-	Iu	(pymeasure.instruments.ametek.Ametek7270 prop- erty), 104
erty), 68	arimantOua	u i d	(pymeasure.instruments.bkprecision.BKPrecision9130B
method), 52	erimeniQue	ивч	property), 115
has_persistent_switch_enabled()	(рутеа-	id	(pymeasure.instruments.danfysik.Danfysik8500 prop-
sure.instruments.ami.AMI430 method	4.0		erty), 117
has_pulse_modulation	(рутеа-	id	(pymeasure.instruments.fluke.Fluke7341 property),
sure.instruments.agilent.Agilent8257			120
erty), 68		id	(pymeasure.instruments.fwbell.FWBell5080 property),
has_supported_version()	(рутеа-		121
sure.adapters.VISAAdapter static	method),		(pymeasure.instruments.Instrument property), 62
37		id	(pymeasure.instruments.keithley.Keithley2000 prop-
header() (pymeasure.experiment.resi	ılts.Results	.,	erty), 129
method), 47	TEGE 0		(pymeasure.instruments.keithley.Keithley2260B prop-
heater (pymeasure.instruments.oxfordinstrum	ents.ITC503		erty), 133
<pre>property), 181 heater_gas_mode</pre>	(	ıu	(pymeasure.instruments.keithley.Keithley2400 prop- erty), 136
sure.instruments.oxfordinstruments.I	(pymea-	iд	(pymeasure.instruments.keithley.Keithley2450 prop-
property), 181	10303		erty), 142
heater_range	(рутеа-	id	(pymeasure.instruments.keithley.Keithley2600 prop-
sure.instruments.lakeshore.LakeShor	4.5		erty), 158
property), 165		id	(pymeasure.instruments.keithley.Keithley2700 prop-
high_freq (pymeasure.instruments.srs.SR570	property),		erty), 147
205	1 1 3//	id	(pymeasure.instruments.keithley.Keithley2750 prop-
high_frequency_resolution	(pymea-		erty), 157
sure.instruments.rohdeschwarz.sfm.S property), 190	TFM	id	(pymeasure.instruments.keithley.Keithley6221 property), 150
	ilent4156.A	gi <b>ile</b> h	###@asure.instruments.keithley.Keithley6517B prop-
property), 78			erty), 155
home() (pymeasure.instruments.anaheimauton method), 108	nation.DPSe	er <b>ie</b> s.	MynneCuma-Mutruments.keysight.KeysightDSOX1102G property), 160
home() (pymeasure.instruments.newport.e. method), 168	sp300.Axis	id	(pymeasure.instruments.keysight.KeysightN5767A property), 163

id	(pymeasure.instruments.signalrecovery.DSP7265 property), 202	internal_frequency sure.instruments.agilent.Agilent8257D	pymea- prop-
id	(pymeasure.instruments.yokogawa.Yokogawa7651	erty), 68	
	property), 215		рутеа-
_	Frame (class in pymeasure.display.widgets), 55	sure.instruments.agilent.Agilent8257D	prop-
Image	Widget (class in pymeasure.display.widgets), 55	erty), 68	
imode	(pymeasure.instruments.signalrecovery.DSP7265 property), 202	<pre>internalfrequency (pymeasure.instruments.srs</pre>	s.SR860
imped	ance (pymeasure.instruments.agilent.AgilentE4980		рутеа-
-	property), 71	sure.instruments.pendulum.cnt91.CNT9	
init_	curve_buffer() (pymea-	property), 184	
	sure.instruments.signalrecovery.DSP7265	* * ·	рутеа-
	method), 202	sure.instruments.srs.SR570 property), 2	
initi			рутеа-
	sure.instruments.agilent.agilentB1500.AgilentB1		<i>I</i>
	method), 93	method), 69	
initi			рутеа-
	sure.instruments.agilent.agilentB1500.AgilentB1		pymea
	method), 93	method), 129	
Tnnut	(class in pymeasure.display.inputs), 50		рутеа-
	_config (pymeasure.instruments.srs.SR830 prop-	sure.instruments.keithley.Keithley2400	рушси
Tiput.	erty), 207	method), 136	
innut	• •		nymaa
Tipu c	**	sure.instruments.keithley.Keithley2450	рутеа-
innu+	property), 207	· · · · · · · · · · · · · · · · · · ·	
Tipu c	_coupling (pymeasure.instruments.srs.SR860	method), 142 is_buffer_full() (	, , , , , , , , , , , , , , , , , , ,
i nnu+	property), 210	· · · · · · · · · · · · · · · · · · ·	рутеа-
Input	_current_gain (pymea-	sure.instruments.keithley.Keithley2700	
·	sure.instruments.srs.SR860 property), 210	method), 147	·
input	_grounding (pymeasure.instruments.srs.SR830	,	рутеа-
	property), 207	sure.instruments.keithley.Keithley6221	
input.	_notch_config (pymea-	method), 150	
	sure.instruments.srs.SR830 property), 207	,	рутеа-
input	_range (pymeasure.instruments.srs.SR860 prop- erty), 210	sure.instruments.keithley.Keithley6517B method), 155	i
input	_shields (pymeasure.instruments.srs.SR860	is_current_stable() (	рутеа-
	property), 210	sure.instruments.danfysik.Danfysik8500	
input	_signal (pymeasure.instruments.srs.SR860 prop-	method), 117	
	erty), 210	$\verb is_enabled  (pymeasure. instruments. a gilent. A gilenter a gi$	ent8257D
input	_voltage_mode	property), 68	
	sure.instruments.srs.SR860 property), 210	is_enabled() (	рутеа-
Input	sWidget (class in pymeasure.display.widgets), 55	sure.instruments.danfysik.Danfysik8500	
insta	nt_voltage_1 (pymea-	method), 117	
	sure. instruments. razor bill. razor bill RP100	is_enabled() (	рутеа-
	property), 185	sure.instruments.keysight.KeysightN576	7A
insta	nt_voltage_2 (pymea-	method), 163	
	sure.instruments.razorbill.razorbillRP100 property), 185	is_moving() (pymeasure.instruments.parker.Par method), 183	kerGV6
Instr	ument (class in pymeasure.instruments), 61	is_out_of_range() (pymeasure.instruments.srs	s.SR830
	erInput (class in pymeasure.display.inputs), 51	method), 207	
	erParameter (class in pymea-	is_ready()(pymeasure.instruments.danfysik.Da	nfvsik8500
9	sure.experiment.parameters), 44	method), 117	5,5
intea	ration_time (pymea-	is_running() (pymeasure.display.manager.M	<b>l</b> anager
9	sure.instruments.agilent.agilent4156.Agilent4156		
	property), 79		рутеа-

sure.instruments.danfysik.Danfysik8500 method), 117	LDCPolarity (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 214
is_set() (pymeasure.experiment.parameters.Parameter method), 46	LDCStatus (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 214
ITC503 (class in pymea-	left_limit(pymeasure.instruments.newport.esp300.Axis
sure.instruments.oxfordinstruments), 180	property), 168
J	level (pymeasure.instruments.rohdeschwarz.sfm.SFM property), 190
<pre>join() (pymeasure.display.thread.StoppableQThread</pre>	level_lin(pymeasure.instruments.anritsu.AnritsuMS9710C
method), 54	property), 111
join() (pymeasure.experiment.workers.Worker method), 46	level_log(pymeasure.instruments.anritsu.AnritsuMS9710C property), 111
40	level_mode (pymeasure.instruments.rohdeschwarz.sfm.SFM
K	property), 190
Keithley2000 (class in pymea-	level_opt_attn (pymea-
sure.instruments.keithley), 126	sure.instruments.anritsu.AnritsuMS9710C
Keithley2260B (class in pymea-	property), 111
sure.instruments.keithley), 132	level_scale(pymeasure.instruments.anritsu.AnritsuMS9710C
Keithley2400 (class in pymea-	property), 111
sure.instruments.keithley), 134	lia_status (pymeasure.instruments.srs.SR830 prop-
Keithley2450 (class in pymea-	erty), 207
sure.instruments.keithley), 140	line_frequency (pymea-
Keithley2600 (class in pymea-	sure.instruments.keithley.Keithley2400 prop-
sure.instruments.keithley), 158	erty), 136
Keithley2700 (class in pymea-	line_frequency_auto (pymea-
sure.instruments.keithley), 146	sure.instruments.keithley.Keithley2400 prop-
Keithley2750 (class in pymea-	erty), 136
sure.instruments.keithley), 157	LINEAR (pymeasure.instruments.agilent.agilentB1500.SamplingMode attribute), 102
Keithley6221 (class in pymea-	LINEAR_DOUBLE (pymea-
sure.instruments.keithley), 148	sure.instruments.agilent.agilentB1500.SweepMode
Keithley6517B (class in pymea-	attribute), 102
sure.instruments.keithley), 153 KeysightDSOX1102G (class in pymea-	LINEAR_SINGLE (pymea-
KeysightDSOX1102G (class in pymea- sure.instruments.keysight), 158	sure.instruments.agilent.agilentB1500.SweepMode
KeysightN5767A (class in pymea-	attribute), 102
sure.instruments.keysight), 163	<pre>list_resources() (in module pymeasure.instruments),</pre>
kill() (pymeasure.instruments.parker.ParkerGV6	65
method), 183	Listener (class in pymeasure.experiment.listeners), 42
-	ListInput (class in pymeasure.display.inputs), 51
L	ListParameter (class in pymea-
labels() (pymeasure.experiment.results.Results	sure.experiment.parameters), 45
method), 47	load() (pymeasure.display.manager.Manager method),
LakeShore331 (class in pymea-	53
sure.instruments.lakeshore), 165	load() (pymeasure.display.widgets.ImageWidget
LakeShore425 (class in pymea-	method), 55
sure.instruments.lakeshore), 166	load() (pymeasure.display.widgets.PlotWidget method), 55
LakeShoreUSBAdapter (class in pymea-	load() (pymeasure.display.widgets.TabWidget method),
sure.instruments.lakeshore), 164	56
LDCCurrent (pymeasure.instruments.thorlabs.ThorlabsPra	p8000   10ad() (pymeasure.experiment.results.Results static
property), 214	method), 47
LDCCurrentLimit (pymea-	load_sequence() (pymea-
sure.instruments.thorlabs.ThorlabsPro8000	sure.display.widgets.SequencerWidget method),
property), 214	56

local() (pymeasure.instruments.danfysik.Danfysik8500 method), 117	ManagedImageWindow sure.display.window	(class in vs), 57	рутеа-	
local() (pymeasure.instruments.keithley.Keithley2000 method), 129	ManagedWindow (class in py	vmeasure.displ	lay.windows),	
LOG_10 (pymeasure.instruments.agilent.agilentB1500.Sampattribute), 102	p <b>liagMged</b> WindowBase ( sure.display.window	class in	рутеа-	
LOG_100 (pymeasure.instruments.agilent.agilentB1500.San attribute), 103	- ·	display.manag		ode
LOG_25 (pymeasure.instruments.agilent.agilentB1500.Sampattribute), 102				
LOG_250 (pymeasure.instruments.agilent.agilentB1500.San attribute), 103				
LOG_50 (pymeasure.instruments.agilent.agilentB1500.Sampattribute), 102			_	
LOG_5000 (pymeasure.instruments.agilent.agilentB1500.Sa attribute), 103	amplingMo <b>pe</b> operty), 122 max_current(pymeasure.ins	truments.delta	elektronika.SM70	)45D
LOG_DOUBLE (pymeasure.instruments.agilent.agilentB1500 attribute), 102	.SweepModeroperty), 119 max_current(pymeasure.ins	truments.keith	ley.Keithley2400	
log_magnitude() (pymea- sure.instruments.agilent.Agilent8722ES	<pre>property), 136 max_current (pymeasure.ins</pre>	truments.keith	ley.Keithley2450	
method), 69 log_ratio(pymeasure.instruments.signalrecovery.DSP72 property), 202	property), 142 @max_frequency (pymeasure property), 122	e.instruments.h	р.НР33120А	
LOG_SINGLE (pymeasure.instruments.agilent.agilentB1500 attribute), 102		e.instruments.h	р.НР33120А	
LogHandler (class in pymeasure.display.log), 52 LogHandler.Emitter (class in pymeasure.display.log), 52	max_resistance sure.instruments.ket erty), 136	thley.Keithley2	(pymea- 2400 prop-	
LogWidget (class in pymeasure.display.widgets), 55 low_freq (pymeasure.instruments.srs.SR570 property), 205	max_resistance sure.instruments.ket erty), 142	thley.Keithley2	(pymea- 2450 prop-	
low_freq_out_amplitude (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 68	max_voltage(pymeasure.ins property), 119 max_voltage(pymeasure.ins			)45D
low_freq_out_source (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 68	property), 136 max_voltage (pymeasure.ins property), 142			
lower_sideband_enabled (pymea- sure.instruments.rohdeschwarz.sfm.SFM	maximums (pymeasure.instrum property), 136			
property), 191	maximums (pymeasure.instrum property), 142			. M . C
mag (pymeasure.instruments.ametek.Ametek7270 prop-	maxspeed(pymeasure.instrum property), 108 mean_current	nents.ananetm	ашотаноп.DPSe (рутеа-	riesmotorCor
erty), 104 mag (pymeasure.instruments.signalrecovery.DSP7265 property), 202	sure.instruments.ket erty), 136	thley.Keithley2		
magnet_current (pymeasure.instruments.ami.AMI430 property), 106	mean_current sure.instruments.ket	thley.Keithley2	(pymea- 2450 prop-	
magnitude (pymeasure.instruments.srs.SR830 property), 207	<pre>erty), 142 mean_resistance</pre>		(рутеа-	
$\begin{tabular}{ll} {\tt magnitude} \ (py measure. instruments. srs. SR 860 \ property), \\ 211 \end{tabular}$	sure.instruments.ket erty), 136	thley.Keithley2		
<pre>magnitude() (pymeasure.instruments.agilent.Agilent8722</pre>	mgean_resistance sure.instruments.ket	thley.Keithley2	(pymea- 2450 prop-	

property), 136 method), 155		erty), 142				measure	_current	()	(рутеа-
ean_voltage	mean_vo	oltage			(рутеа-		sure.instr	uments.keithley.Keithley240	00
means voltage (pymeasure.instruments.keithley.Keithley2450 property), 142 (pymeasure.instruments.keithley.Keithley2450 property), 143 (pymeasure.instruments.keithley.Keithley2450 property), 145 (pymeasure.instruments.keithley.Keithley2450 property), 146 (pymeasure.instruments.agilent.a		sure.instru	ments.keithley. <b>K</b>	Keithley2400	prop-				
sure.instruments.keithley.Keithley2450 property), 126 (gymeasure.instruments.keithley.Keithley2450 property), 126 meass_mode() (pymeasure.instruments.agilent.agilent.agilentB1500.sMU property), 127 meas_range_current_() (pymeasure.instruments.agilent.agilentB1500.sMU property), 128 meas_p_wode() (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_current_() (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 measure_voltage (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 measure_voltage (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 measure_voltage (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 measure_voltage (pymeasure.instruments.agilentB1500.sMU property), 98 measure_voltage (pymeasure.instruments.agilentB1500.sMU property), 100 measure_voltage (pymeasure.instruments.agilentB1500.sMU property), 100 measure_voltage (pymeasure.instruments.agilentB1500.smu property), 100 measure_voltage (pymeasure.instruments.agilentB1500.smu property), 125 measure_voltage (pymea		erty), 136				measure	_current	()	(рутеа-
means property), 142 (pymeasure_instruments_keithley_Keithley2400 property), 125 (pymeasure_instruments_keithley_Keithley2450 property), 125 (pymeasure_instruments_keithley_Keithley2450 property), 125 (pymeasure_instruments_keithley_Keithley2450 property), 125 (pymeasure_instruments_keithley_Keithley2450 property), 125 (pymeasure_instruments_keithley_Keithley2000 property), 125 (pymeasure_instruments_keithley_Keithley2000 property), 125 (pymeasure_instruments_keithley_Keithley2000 property), 126 (pymeasure_instruments_keithley_Keithley2000 property), 127 (pymeasure_instruments_keithley_Keithley2000 property), 128 (pymeasure_instruments_keithley_Keithley2000 property), 129 (pymeasure_instruments_keithley_Keithley2000 property), 125 (pymeasure_instruments_keithley_Keithley2400	mean_vo	oltage			(pymea-		sure.instr	uments.keithley.Keithley245	50
means (pymeasure.instruments.keithley.Keithley2400 property), 126 method), 155 method), 155 method), 155 measure_outstruments.agilent.agilentB1500.sMU property), 126 meas_op_mode (pymeasure.instruments.agilent.agilentB1500.sMU property), 97 meas_range_current_out (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_outrent_aut o() (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_outrent_aut o() (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_outrent_aut o() (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_outrent_agilentB1500.sMU property), 98 meas_range_outrent_agilentB1500.sMU property), 98 meas_range_outrent_agilent.agilentB1500.sMU property), 98 meas_range_outrent_sagilent.agilentB1500.sMU property), 98 meas_range_outrent_sagilent.agilentB1500.sMU property), 98 measure_instruments.agilent.agilentB1500.sMU property), 109 measure_instruments.agilent.agilentB1500.sMU property), 100 measure_instruments.agilent.agilentB1500.sMU property, 125 measure_instruments.agilent.agilentB1500.sMU property, 125 measure_instruments.agilent.agilentB1500.sMU property, 125 measure_resistance() (pymeasure_instruments.agilent.agilentB1500.sMU property), 125 measure_resistance() (pymeasure_instruments.agilent.agilentB1500.sMU property), 125 measure_resistance() (pymeasure_instruments.agilent.agilentB1500.sMU property), 125 measure_resistance() (pymeasure_instru		sure.instru	ments.keithley. <mark>K</mark>	Keithley2450	prop-		method),	142	
means (pymeasure.instruments.keithley.Keithley2450 means_node() (pymeasure.instruments.agilent.agilentB1500.sMU property), 142 meas_node() (pymeasure.instruments.agilent.agilentB1500.sMU property), 98 meas_range_current withouthouthouthouthouthouthouthouthouthou		erty), 142				measure	_current	()	(рутеа-
means (pymeasure.instruments.kp.HP3478A property), 142 property), 142 property), 142 property), 142 property), 142 property), 143 property), 144 property), 145 property), 145 property), 146 measure_Current_auto() (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 meas_range_Current_auto() (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 measure_Sure.instruments.agilent.agilentB1500.SMU property), 98 measure_woltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 measure_woltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 measure_woltage (pymeasure.instruments.agilentB1500.SMU property), 102 measure_woltage (pymeasure.instruments.agilentB1500.SMU property), 102 measure_woltage (pymeasure.instruments.agilentB1500.SMU property), 110 measure_peak() (pymeasure.instruments.agilentB1500.SMU property), 125 measure_woltage (pymeasure.instruments.agilentB1500.SMU property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_contrinuity() (pymeasure.instruments.hp.HP3478A property), 125 measure_contrinuity() (pymeasure.instruments.hp.HP3478A property), 125 measure_contrinuity() (pymeasure.instruments.hp.HP3478A property), 125 measure_contrinuity() (pymeasure.instrum	means	(pymeasi	ıre.instruments.l	keithley.Keith	aley2400		sure.instr	uments.keithley.Keithley651	7B
property), 142 meas_mode() (pymeasure.instruments.agilent.agilentB1500magilantB1500		property),	136				method),	155	
meas_mode() (pymeasure.instruments.agilent.agilentB1500m4gstart&Dt00 property), 125 measure() (pymeasure.instruments.agilent.agilentB1500.SMU property), 125 measure() (pymeasure.instruments.agilent.agilentB1500.SMU property), 125 measure() (pymeasure.instruments.agilent.agilentB1500), 102 measure() (pymeasure.instruments.bp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_concurrent_functions (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurrent_functions (pymeasure.instruments.hp.HP3478A property), 125 measure_concurrent_functions (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_concurrent_functions (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurrent_functions (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurrent_functions (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurrent_functions (pymeasure.instruments.hp.HP	means			keithley.Keith	aley2450	measure		= :	.HP3478A
method), 94 meas_o_mode (pymeasure_instruments.agilent.agilentB1500.SMU property), 97 meas_range_current (pymeasure_instruments.agilent.agilentB1500.SMU property), 98 meas_range_current_auto() (pymeasure_instruments.agilent.agilentB1500.SMU method), 98 meas_range_voltage (pymeasure_instruments.agilent.agilentB1500.SMU property), 98 MeasMode (class in pymeasure_instruments.agilent.agilentB1500), 102 measure_bolde (class in pymeasure_instruments.agilent.agilentB1500), 102 measure_bolde (class in pymeasure_instruments.agilent.agilentB1500), 102 measure_bolde (class in pymeasure.instruments.agilent.agilentB1500), 102 measure_bolde (class in pymeasure.instruments.agilentB1500), 102 measure_bolde (class in pymeasure.instruments.agilent.agilentB1500), 102 measure_logophode (class in pymeasure.instruments.agilent.agilentB1500), 102 measure_cologophode (class in pymeasure.instruments.agilent.agilentB1500), 102 measure_logophode (class in pymeasure.instruments.agilent.agilent	meas_mo			its.agilent.ag	ilentB150	Onegistenne			.HP3478A
meas_op_mode						Ü		= :	
sure.instruments.agilent.agilentB1500.SMU property), 97  meas_range_current sure.instruments.agilent.agilentB1500.SMU property), 98  meas_range_current_auto() (pymeasure.instruments.agilent.agilentB1500.SMU method), 98  meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98  meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98  measure_instruments.agilent.agilentB1500.SMU property), 98  measure_instruments.agilent.agilentB1500.NU property), 102  measure_sexperiment.parameters), 45 measure_clopymeasure.instruments.agilent.agilentB1500, 102 measure_() (pymeasure.instruments.agilent.agilentB1500, 102 measure_() (pymeasure.instruments.lakeshore.LakeShore225 method), 199 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.keithley.Keithley2400 method), 129 measure_resistance() (pymeasure.instruments.keithley.Keithley2400 method), 136 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2450 measure_resistance() (pymeasure.instruments.keithley.Keithley2450 method), 129 measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125 measure_resistance() (pymeasure.instruments.hp.HP3478A property), 126 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2450 method), 129 measure_resistance() (pymeasure.instruments.keithley.Keithl	meas_op				(рутеа-	measure			(рутеа-
meas_range_current (pymeasure_instruments.agilent.agilentB1500.SMU property), 98 meas_range_current_auto() (pymeasure_instruments.agilent.agilentB1500.SMU method), 98 meas_range_voltage (pymeasure_instruments.agilent.agilentB1500.SMU property), 98 meass_range_voltage (pymeasure_instruments.agilent.agilentB1500.SMU property), 98 measure_instruments.agilent.agilentB1500, 102 method), 199 measure_current_order sure_instruments.agilent.agilentB1500), 102 method), 190 measure_cyneasure_instruments.agilent.agilentB1500), 102 method), 190 measure_cyneasure_instruments.agilent.agilentB1500, 102 method), 102 method), 190 measure_cyneasure_instruments.agilentB1500, 102 method), 102 method), 190 measure_cyneasure_instruments.agilent.agilentB1500, 102 method), 102 method), 190 measure_cyneasure_instruments.agilentB1500, 102 method), 102 method), 129 measure_cyneasure_instruments.agilentB1500, 102 method), 129 measure_cyneasure_instruments.agilent			ıments.agilent.aş	gilentB1500.	SMU		sure.instr	uments.keithley.Keithley200	00
sure.instruments.agilent.agilentB1500.SMU property), 98 meas_range_current_auto() (pymeasure.instruments.agilent.agilentB1500.SMU method), 98 meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98 measure (class in pymeasure.instruments.agilent.agilentB1500), 102 measure (class in pymeasure.instruments.agilent.agilentB1500), 102 measure (class in pymeasure.instruments.agilent.agilentB1500), 102 measure (class in pymeasure.experiment.parameters), 45 measure (c) (pymeasure.instruments.agilent.agilent4156.Agilent4156 property), 125 method), 166 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_ACV (pymeasure.instruments.hp.HP3478A property), 125 measure_Capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurent (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_concurent (pymeasure.instruments.keithley.Keithl									
meas_range_current_auto() (pymeasure.instruments.agilent.agilentB1500.SMU method), 98  meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98  MeasMode (class in pymeasure.instruments.agilent.agilentB1500), 102 measure2 (class in pymeasure.instruments.agilent.agilentB1500), 102 measure2 (pymeasure.instruments.agilent.agilentB1500), 102 method), 129  Measurable (class in pymeasure.instruments.agilent.agilentB1500), 102 method), 79  measure () (pymeasure.instruments.agilent.agilentB1500), 102 method), 79  measure () (pymeasure.instruments.agilent.agilentB1500), 102 method), 199  measure () (pymeasure.instruments.agilent.agilentB1500), 102 method), 199  measure () (pymeasure.instruments.agilent.agilentB1500), 102 method), 199  measure () (pymeasure.instruments.agilent.agilentB1500), 102 measure () (pymeasure.instruments.bp.HP3478A property), 125 measure () (pymeasure.instruments.bp.HP3478A property), 125 measure () (pymeasure.instruments.keithley.Keithley2000 method), 136 measure () (pymeasure.instruments.keithley.Keithley2400 method), 129 measure () (pymeasure.instruments.keithley.Keithley2400 method), 129 measure () (pymeasur	meas_ra	ange_curre	ent		(рутеа-	measure	_frequen	cy()	(рутеа-
meas_range_current_auto() (pymeasure.instruments.agilent.agilentB1500.SMU method), 98  meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98  MeasMode (class in pymeasure.instruments.agilent.agilentB1500), 102  measUpMeasurable (class in pymeasure.instruments.agilent.agilentB1500), 102  measureO(pymeasure.instruments.agilent.agilentB1500), 102  measureO(pymeasure.instruments.agilent.agilentB1500), 102  measureO(pymeasure.instruments.agilent.agilentB1500), 102  measureO(pymeasure.instruments.agilent.agilentB1500), 102  measureO(pymeasure.instruments.agilent.agilentB1500), 102  measureO(pymeasure.instruments.agilent.agilentB1500, 102  method), 190  measure_ACI (pymeasure.instruments.hp.HP3478A property), 125  measure_ACI (pymeasure.instruments.hp.HP3478A property), 125  measure_ACI (pymeasure.instruments.hp.HP3478A property), 125  measure_Concurent_functions (pymeasure.instruments.keithley.Keithley2400 method), 136  measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 method), 136  measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 method), 129  measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 method), 136  measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 method), 129  measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125  measure_resistance() (pymeasure.instruments.keithley.Keithley2400 method), 136  measure_resistance() (pymeasure.instruments.keithley.Keithley2400 method), 129  measure_resistance() (pymeasure.instruments.keithley.Keithley2000 method), 129  measure_resistance() (pymeasure.				gilentB1500.	SMU				00
sure.instruments.agilent.agilentB1500.SMU method), 98  meas_range_voltage (pymeasure.instruments.agilent.agilentB1500.SMU property), 98  MeasMode (class in pymeasure.instruments.agilent.agilentB1500), 102  MeasOpMode (class in pymeasure.instruments.agilent.agilentB1500), 102  Measurable (class in pymeasure.experiment.parameters), 45  measure() (pymeasure.instruments.agilent.agilentA156.Agilent4156 property), 125  method), 79  measure() (pymeasure.instruments.lakeshore.LakeShore425  measure_ACT (pymeasure.instruments.hp.HP3478A property), 125  measure_ACT (pymeasure.instruments.hp.HP3478A property), 125  measure_ACT (pymeasure.instruments.hp.HP3478A property), 125  measure_ACT (pymeasure.instruments.hp.HP3478A property), 125  measure_Concurent_functions (pymeasure.instruments.keithley.Keithley2000 method), 139  measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136  measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125  measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125  measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125  measure_resistance() (pymeasure.instruments.keithley.Keithley2000 method), 129  measure_resistance() (pymeasure.instruments.keithley.Keithley2000 method), 129  measure_resistance() (pymeasure.instruments.keithley.Keithley2000 method), 129  measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125  measure_resistance() (pymeasure.instruments.keithley.Keithley2000 method), 129  measure_resistance() (pymeasure.instruments.hp.HP3478A property), 125  measure_resista	meas_ra				(рутеа-	measure			(рутеа-
meas_range_voltage		_		gilentB1500.	SMU		sure.instr	uments.anritsu.AnritsuMS9	
meas_range_voltage		method), 9	98				property)	, 111	
measure () (pymeasure.instruments.agilent.agil	meas_ra	ange_volta	age		(рутеа-	measure			(рутеа-
MeasMode (class in pymeasure.instruments.agilent.agilentB1500), 102 measOpMode (class in pymeasure.instruments.agilent.agilentB1500), 102 method), 103 measure() (pymeasure.instruments.agilent.agilent.agilent4156.Agilent4156 property), 125 method), 166 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_Capacity() (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A property), 125 method), 114 measure_concurent_functions sure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.continuity() measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 129 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2400 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 method), 129 measure.instruments.keithley.Keithley2000		sure.instru	ıments.agilent.aş	gilentB1500.	SMU		sure.instr	uments.anritsu.AnritsuMS9	710C
sure.instruments.agilent.agilentB1500), 102 measOpMode (class in pymeasure.instruments.agilent.agilentB1500), 102 measurable (class in pymeasure.experiment.parameters), 45 measure() (pymeasure.instruments.agilent.agilent.agilentB1500, 102 method), 79 measure() (pymeasure.instruments.agilent.agilent.agilentB150AgilentB150By, 125 method), 19 measure() (pymeasure.instruments.lakeshore.LakeShore425 method), 166 measure_ACI (pymeasure.instruments.hp.HP3478Aproperty), 125 measure_ACI (pymeasure.instruments.hp.HP3478Aproperty), 125 measure_ACV (pymeasure.instruments.hp.HP3478Aproperty), 125 measure_Capacity() (pymeasure.instruments.hp.HP3478Aproperty), 125 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400property), 129 measure_continuity() (pymeasure.instruments.keithley.Keithley2400property), 129 measure_continuity() (pymeasure.instruments.keithley.Keithley2400property), 125 measure_continuity() (pymeasure.instruments.keithley.Keithley2400property), 125 measure_continuity() (pymeasure.instruments.keithley.Keithley2400property), 125 measure_continuity() (pymeasure.instruments.keithley.Keithley2400property), 125 measure_resistance() (pymeasure.instruments.keithley.Keithley2450property), 125 measure_resistance()		property),	98				method),	111	
MeasOpMode (class in pymea- sure.instruments.agilent.agilentB1500), 102 measurable (class in pymea- sure.experiment.parameters), 45 measure() (pymeasure.instruments.agilent.agilent4156.Agilent4156 property), 125 method), 79 measure() (pymeasure.instruments.agilent.agilent4156.Agilent4156 property), 125 method), 166 measure() (pymeasure.instruments.lakeshore.LakeShore425 measure() (pymeasure.instruments.lakeshore.LakeShore425 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_CV (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.capacity() (pymeasure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2450 measure_resistance() (pymeasure.instruments.keithley.Keit	MeasMod	de	(class	in	рутеа-	measure	_period(	)	(рутеа-
Measurable (class in pymeasure.experiment.parameters), 45 measure—R4W (pymeasure.instruments.hp.HP3478A property), 125 measure—resistance() (pymeasure.instruments.hp.HP3478A property), 125 measure—R5 wire.instruments.hp.HP3478A property), 125 measure—resistance() (pymeasure.instruments.hp.HP3478A property), 125 measure—resistance() (pymeasure.instruments.hp.HP3478A property), 125 measure—resistance() (pymeasure.instruments.hp.HP3478A property), 125 measure—resistance() (pymeasure.instruments.hp.HP3478A property), 125 measure—resistance() (pymeasure.instruments.keithley.Keithley2400 method), 136 measure—resistance() (pymeasure.instruments.keithley.Keithley2400 method), 136 measure—resistance() (pymeasure.instruments.keithley.Keithley2450 method), 143 measure—resistance() (pymeasure.instruments.keithley.Keithley2450 method), 143 measure—resistance() (pymeasure.instruments.keithley.Keithley2450 method), 143 measure—resistance() (pymeasure.instruments.keithley.Keithley2450 method), 155 measure—resistance() (pymeasure.instruments.keithley.Keithley2450 method), 129 measure—resistance() (pymeasure.instruments.keithley.Keithley2400 method), 129 measure—resistance() (pymeasure.instruments.keithley.Keithley2400 method), 129 measure—resistance() (pymeasure.instruments.keithley.Keithley2450 method), 129 measure—resistance() (		sure.instru	ıments.agilent.aş	gilentB1500)	, 102		sure.instr	uments.keithley.Keithley200	00
Measurable (class in pymea- sure.experiment.parameters), 45 measure_R4W (pymeasure.instruments.hp.HP3478A measure() (pymeasure.instruments.agilent.agilent4156.Agilent4156 property), 125 method), 79 measure_resistance() (pymea- measure_ACI (pymeasure.instruments.hp.HP3478A measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_resistance() (pymea- sure.instruments.attocube.anc300.Axis method), 114 measure_capacity() (pymea- sure.instruments.keithley.Keithley2400 prop- erty), 136 measure_resistance() (pymea- sure.instruments.keithley.Keithley2400 prop- erty), 136 measure_resistance() (pymea- sure.instruments.keithley.Keithley2400 prop- erty), 136 measure_resistance() (pymea- sure.instruments.keithley.Keithley2450 measure_resistance(	MeasOpM	<b>l</b> ode	(class	in	рутеа-		method),	129	
sure.experiment.parameters), 45 measure() (pymeasure.instruments.agilent.agilent.4156.Agilent4156 property), 125 method), 79 measure() (pymeasure.instruments.lakeshore.LakeShore425 method), 166 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_ACV (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 measure_continuity() (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_continuity() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_continuity() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_continuity() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_voltage() (pymeasure.instruments.hp.HP3478A measure_voltage() (pymeasure.instruments.hp.HP3478A measure_voltage() (pymeasure.instruments.deltaelektronika.SM7045D measure_voltage() (pymeasure.instruments.deltaelektronika.SM7045D		sure.instru	ıments.agilent.aş	gilentB1500)	, 102	measure	_R2W	(pymeasure.instruments.hp	.HP3478A
measure() (pymeasure.instruments.agilent.agilent4156.Agilent4156 property), 125 method), 79 measure() (pymeasure.instruments.lakeshore.LakeShore425 method), 166 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_CACV (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.hp.HP3478A method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_voltage (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_current() (pymeasure.instruments.keithley.Keithley2000 measure_voltage (pymeasure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure.voltage() (pymeasure.instruments.keithley.Keithley2000 measure.voltage() (pymeasure.instruments.keithley.Keithley2000 measure.voltage() (pymeasure.instruments.keithley.Keithley2000 measure.instruments.keithley.Keithley2000 measure.instruments.keithley	Measura	able	(class	in	рутеа-		property)	, 125	
method), 79 measure() (pymeasure.instruments.lakeshore.LakeShore425 sure.instruments.keithley2000 method), 166 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.capacity() (pymeasure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.continuity() (pymeasure.		sure.exper	iment.parametei	s), 45		measure	_R4W	(py measure. instruments. hp	.HP3478A
measure() (pymeasure.instruments.lakeshore.LakeShore425 sure.instruments.keithley.12000 method), 166 method), 129 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 sure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure_continuity() (pymeasure_continuity() (pymeasure_continuity() (pymeasure_continuity() (pymeasure_instruments.keithley.Keithley2000 measure_current (pymeasure_instruments.keithley.Keithley2000 measure_current (pymeasure_instruments.keithley.Keithley2000 measure_current() (pymeasure_instruments.keithley.Keithley2000 measure_current() (pymeasure_instruments.keithley.Keithley2000 measure_voltage (pymeasure_instruments.keithley.Keithley2000 property), 119 measure_voltage() (pymeasure_voltage() (pymeasure	measure	e() (pymeas	sure.instruments.	agilent.agile	nt4156.A	gilent4156	property)	, 125	
method), 166 measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 measure_ACV (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_current() (pymeasure.voltage() (pymeasure.vol		method), 7	79			measure			
measure_ACI (pymeasure.instruments.hp.HP3478A property), 125 sure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_voltage() (pymeasure_voltage() (pymeasure_voltage()) (pymeasure_voltage())	measure	e() (pymeas	sure.instruments.	lakeshore.La	akeShore4	!25	sure.instr	uments.keithley.Keithley200	00
property), 125 measure_ACV (pymeasure.instruments.hp.HP3478A property), 125 measure_capacity() (pymeasure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 125 measure_nestruments.keithley.Keithley2400 property), 125 measure_nestruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_voltage (pymeasure.instruments.deltaelektronika.SM7045D measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure_resistance() (pymeasure.instruments.keithley.Keithley2450 measure_resistance() (pymeasure.instruments.keithley.Keithley2600 measure_resistance() (pymeasure.instruments.keithley.Keithley2600 measure_resistance() (pymeasure.instruments.keithley.Keithley2600 measure_resistance() (pymeasure.instrument		, ,							
measure_ACV (pymeasure.instruments.hp.HP3478A property), 125 measure_resistance() (pymeasure.instruments.attocube.anc300.Axis method), 143 method), 143 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_temperature() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_temperature() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_voltage (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_voltage() (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_voltage() (pymeasure.instruments.deltaelektronika.SM7045D property), 119 measure_voltage()	measure			ruments.hp.H	<i>IP3478A</i>	measure			
property), 125 measure_capacity() (pymeasure_instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymeasure_instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure_continuity() (pymeasure_instruments.keithley.Keithley2000 measure_current (pymeasure_current (pymeasure_instruments.deltaelektronika.SM7045D measure_current() (pymeasure_instruments.keithley.Keithley2000 measure_current() (pymeasure_instruments.deltaelektronika.SM7045D property), 119 sure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure_voltage() (pymeasure_voltage()) (pymeasure_voltage()) (pymeasure_voltage()) (pymeasure_voltage()) (pymeasure_voltage()) (pymeasure_voltage())									00
measure_capacity() (pymea- sure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions (pymea- sure.instruments.keithley.Keithley2400 prop- erty), 136 measure_continuity() (pymea- sure.instruments.keithley.Keithley2400 prop- measure_continuity() (pymea- sure.instruments.keithley.Keithley2000 measure_current (pymea- sure.instruments.deltaelektronika.SM7045D measure_current() (pymea- sure.instruments.keithley.Keithley2000 measure_current() (pymea- sure.instruments.keithley.Keithley2000 measure_voltage (pymea- sure.instruments.keithley.Keithley2000 measure_voltage() (pymea- sure.instruments.keithley.Keithley2000 (pymea- sure.instruments.keithley.Keithley2000 measure_voltage() (pymea- sure.instruments.keithley.Keithley2450 measure_resistance() (pymea- sure.instruments.keithley.Keithley2500 measure_resistance() (pymea- sure.instruments.keithley.Keithley2517B measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() (pymea- sure.instruments.keithley.Keithley2000 measure_continuity() (pymea- sure.instruments.keithley.Keithley2000 measure_resistance() (pymea- sure.instruments.keithley.Keithley517B measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() (pymea- sure.instruments.keithley.Keithley2000 measure_temperature() (pymea- sure.instruments.keithley.Keithley2000 measure_temperature() (pymea- sure.instruments.keithley.Keithley2000 measure_voltage() (pymea- property), 119 measure_voltage() (pymea- sure.instruments.keithley2500	measure			uments.hp.H	IP3478A				
sure.instruments.attocube.anc300.Axis method), 114 measure_concurent_functions sure.instruments.keithley.Keithley2400 prop- erty), 136 measure_continuity() measure_continuity() sure.instruments.keithley.Keithley2000 measure_current (pymea- method), 155 measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() sure.instruments.keithley.Keithley2000 measure_current (pymea- property), 119 measure_current() sure.instruments.keithley.Keithley2000 measure_current() sure.instruments.keithley.Keithley2000 measure_voltage property), 119 measure_voltage() measure_voltage() measure_voltage() property), 119 measure_voltage() measure_voltage()						measure			* *
method), 114 measure_concurent_functions (pymeasure.instruments.keithley.Keithley2400 property), 136 measure_continuity() (pymeasure.instruments.keithley.Keithley2000 method), 129 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.keithley.Keithley2000 measure_voltage (pymeasure.instruments.deltaelektronika.SM7045D measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.keithley.Keithley2000)	measure				(рутеа-				50
measure_concurent_functions sure.instruments.keithley.Keithley2400 prop- erty), 136 measure_continuity() sure.instruments.keithley.Keithley2000 method), 129 measure_current sure.instruments.keithley.Keithley2000 measure_current sure.instruments.keithley.Keithley2000 measure_current sure.instruments.keithley.Keithley2000 measure_current() sure.instruments.keithley.Keithley2000 measure_current() sure.instruments.keithley.Keithley2000 measure_voltage sure.instruments.deltaelektronika.SM7045D property), 119 sure.instruments.keithley.Keithley2000 measure_voltage property), 119 measure_voltage() sure.instruments.keithley.Keithley6517B measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() (pymeasure.instruments.keithley.Keithley2000 measure_temperature() sure.instruments.keithley.Keithley2000 measure_voltage sure.instruments.keithley.Keithley6517B measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() sure.instruments.keithley.Keithley2000 measure_temperature() sure.instruments.keithley.Keithley6517B measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 measure_temperature() sure.instruments.keithley.Keithley2000 measure_temperature() sure.instruments.keithley.Keithley2000 measure_voltage property), 119 sure.instruments.keithley.Keithley2000 measure_voltage() sure.instruments.keithley.Keithley2000 measure_voltage() sure.instruments.keithley.Keithley2000 property), 119 sure.instruments.keithley.Keithley2000				anc300.Axis					
sure.instruments.keithley.Keithley2400 property), 136 measure_Rext (pymeasure.instruments.hp.HP3478A property), 125 sure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current), 119 sure.instruments.keithley.Keithley2000 measure_current() (pymeasure.instruments.keithley.Keithley2000 measure_current() (pymeasure.instruments.keithley.Keithley2000 measure_voltage (pymeasure.instruments.deltaelektronika.SM7045D property), 119 sure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.keithley.Keithley2000 measure_voltage()						measure			* *
erty), 136 measure_continuity() (pymeasure.instruments.hp.HP3478A measure_continuity() (pymeasure.instruments.keithley.Keithley2000 measure_current (pymeasure.instruments.keithley.Keithley2000 measure_current() (pymeasure.instruments.deltaelektronika.SM7045D measure_current() (pymeasure.instruments.keithley.Keithley2000 measure_current() (pymeasure.instruments.keithley.Keithley2000 measure_voltage (pymeasure.instruments.deltaelektronika.SM7045D property), 119  sure.instruments.keithley.Keithley2000 measure_voltage() (pymeasure.instruments.hp.HP3478A  measure_Rext (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.keithley.Keithley2000  measure_current() (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.keithley.Keithley2000  measure_current() (pymeasure.instruments.keithley.Keithley2000  measure_voltage (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.keithley.Keithley2000  measure_voltage (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.hp.HP3478A  property), 125 measure_temperature() (pymeasure.instruments.keithley2000  measure_voltage (pymeasure.instruments.keithley2000)  measure_voltage (pymeasure.instruments.keithley2000)  measure_voltage (pymeasure.instruments.hp.HP3478A	measure				4 .				7B
measure_continuity() (pymea-sure.instruments.keithley.Keithley2000 measure_temperature() (pymea-sure.instruments.keithley.Keithley2000 measure_current (pymea-sure.instruments.deltaelektronika.SM7045D measure_current() (pymea-sure.instruments.keithley.Keithley2000 measure_current() (pymea-sure.instruments.keithley.Keithley2000 measure_voltage (pymea-property), 119 sure.instruments.keithley.Keithley2000 measure_voltage() (pymea-property), 119 sure.instruments.keithley.Keithley2000 measure_voltage()			ıments.keithley. <b>k</b>	Ceithley2400	prop-				******
$sure.instruments.keithley.Keithley2000 \\ method), 129 \\ measure\_current \\ sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ measure\_current() \\ sure.instruments.deltaelektronika.SM7045D \\ measure\_current() \\ sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ measure\_voltage() \\ (pymeasure\_voltage()) \\ (pymeasure\_vol$		• .			,	measure			.HP34/8A
$method), 129 \\ measure\_current \\ sure.instruments.keithley.Keithley2000 \\ sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ measure\_current() \\ sure.instruments.deltaelektronika.SM7045D \\ measure\_current() \\ sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ sure.instruments.keithley.Keithley2000 \\ measure\_voltage() \\ (pymeasure\_voltage()) \\$	measure		-		(рутеа-				,
measure_current $(pymea-sure.instruments.deltaelektronika.SM7045D)$ measure_voltage $(pymea-property)$ , 119 $(pymea-sure.instruments.deltaelektronika.SM7045D)$ measure_current() $(pymea-sure.instruments.keithley.Keithley2000)$ measure_voltage() $(pymea-property)$ , 119 $(pymea-property)$				Leitniey2000		measure	_		* *
$sure.instruments.deltaelektronika.SM7045D \\ property), 119 \\ measure\_current() \\ sure.instruments.keithley.Keithley2000 \\ measure\_voltage \\ property), 119 \\ property), 119 \\ measure\_voltage() \\ (pymeasure\_voltage()) \\ (p$	m o c	, ,	129		(				<i>i</i> U
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	measure		o to d-141 1		4 -	m 0 0			(
measure_current() (pymea- property), 119 sure.instruments.keithley.Keithley2000 measure_voltage() (pymea-				ronika.SM/(	143 <b>D</b>	measure	_		
sure.instruments.keithley.Keithley2000 measure_voltage() (pymea-	mo 2 0				(m) ====================================				7043 <b>D</b>
* **	measure				үүтеа-	maaciina			(muma a a
				E 11111E y 2000		measure	_		4.

method), 129		mode	(pymeasure.instruments.attocube.c	anc300.Axis
<pre>measure_voltage()</pre>	(рутеа-		property), 114	
sure.instruments.keithley.Keithley240 method), 137	00	mode	(pymeasure.instruments.hp.HP3478A	property),
measure_voltage()	(рутеа-	mode	(pymeasure.instruments.keithley.K	eithlev2000
sure.instruments.keithley.Keithley245		mouc	property), 129	eiiiie y 2000
method), 143	, 0	modul	Lation_degree	(рутеа-
measure_voltage()	(рутеа-	modul	sure.instruments.rohdeschwarz.sfm.	
sure.instruments.keithley.Keithley651			property), 196	Source_Cricinitei
method), 155	,,,	modul	Lation_enabled	(рутеа-
measurement() (pymeasure.instruments.	Instrument		sure.instruments.rohdeschwarz.sfm.	
static method), 62			property), 191	~1 I/I
measurement()	(рутеа-	modul	Lation_enabled	(рутеа-
sure.instruments.keysight.KeysightDS			sure.instruments.rohdeschwarz.sfm.	
static method), 160	,01111020		property), 196	Source_Cricinitei
measurement_event_enabled	(рутеа-	modu]		
sure.instruments.keithley.Keithley622			pymeasure.display.browser,49	
erty), 150	л ргор		pymeasure.display.curves, 49	
measurement_events	(рутеа-		pymeasure.display.inputs, 50	
sure.instruments.keithley.Keithley622			pymeasure.display.listeners, 52	
erty), 150	л ргор		pymeasure.display.log, 52	
measurement_time	(рутеа-	_	pymeasure.display.manager,52	
sure.instruments.pendulum.cnt91.CN			pymeasure.display.manager, 52	
property), 184	1)1		pymeasure.display.thread, 54	
message_waiting()	(рутеа-		pymeasure.display.widgets,54	
sure.experiment.listeners.Listener	method),		pymeasure.display.windows, 57	
42	memou),		pymeasure.experiment.experiment	41
min_amplitude (pymeasure.instruments.hp.l	HP33120A		pymeasure.experiment.listeners,	
property), 122	11 0012011		pymeasure.experiment.parameters	
min_current (pymeasure.instruments.keithley	Keithlev24		pymeasure.experiment.procedure,	
property), 137			pymeasure.experiment.results, 47	
min_current (pymeasure.instruments.keithley	Keithlev24		pymeasure.experiment.workers, 46	
property), 143			pymeasure.instruments, 59	
min_frequency (pymeasure.instruments.hp.l	HP33120A		ymeasure.instruments.advantest	. 65
property), 122			ymeasure.instruments.advantest	
min_offset (pymeasure.instruments.hp.l	HP33120A	•	66	
property), 122		р	ymeasure.instruments.agilent,6	6
min_resistance	(рутеа-		ymeasure.instruments.agilent.a	
sure.instruments.keithley.Keithley240		•	76	,
erty), 137	r	n	ymeasure.instruments.agilent.a	ailentB1500.
min_resistance	(рутеа-	•	101	,
sure.instruments.keithley.Keithley245		n	oymeasure.instruments.ametek, 10	3
erty), 143	r	_	pymeasure.instruments.ami, 105	
min_voltage(pymeasure.instruments.keithley	Keithlev24	_	ymeasure.instruments.anaheimau	tomation,
property), 137	·		107	
min_voltage(pymeasure.instruments.keithley	Keithley24.		oymeasure.instruments.anapico, 1	
property), 143		_	oymeasure.instruments.anritsu, 1	
minimums (pymeasure.instruments.keithley.Ke	ithley2400	_	ymeasure.instruments.attocube,	
property), 137	0.450	_	oymeasure.instruments.bkprecisi	
minimums (pymeasure.instruments.keithley.Ke	ıthley2450	_	oymeasure.instruments.comedi,65	
property), 143		_	oymeasure.instruments.danfysik,	
Mock (class in pymeasure.instruments), 63	4000	p	oymeasure.instruments.deltaelek	tronika,
mode (pymeasure.instruments.agilent.AgilentE-	4980 prop-		119	•
erty) 71		n	ovmeasure instruments edwards 1	70

pymeasure.instruments.fluke, 120	nicam_bit_error_enabled (pymea-
pymeasure.instruments.fwbell, 120	sure.instruments.rohdeschwarz.sfm.SFM
pymeasure.instruments.hp, 122	property), 191
pymeasure.instruments.keithley, 126	nicam_bit_error_rate (pymea-
pymeasure.instruments.keysight, 158	sure.instruments.rohdeschwarz.sfm.SFM
pymeasure.instruments.lakeshore, 164	property), 191
pymeasure.instruments.newport, 167	nicam_carrier_enabled (pymea-
pymeasure.instruments.ni, 168	sure.instruments.rohdeschwarz.sfm.SFM
<pre>pymeasure.instruments.oxfordinstruments,</pre>	property), 191
180	nicam_carrier_frequency (pymea-
pymeasure.instruments.parker,182	sure.instruments.rohdeschwarz.sfm.SFM
pymeasure.instruments.pendulum, 184	property), 191
pymeasure.instruments.razorbill, 185	nicam_carrier_level (pymea-
pymeasure.instruments.rohdeschwarz,186	sure.instruments.rohdeschwarz.sfm.SFM
pymeasure.instruments.signalrecovery, 200	property), 191
pymeasure.instruments.srs, 203	nicam_control_bits (pymea-
<pre>pymeasure.instruments.tektronix, 213</pre>	sure.instruments.rohdeschwarz.sfm.SFM
pymeasure.instruments.thorlabs, 213	property), 191
<pre>pymeasure.instruments.validators,63</pre>	$\verb nicam_data  (pymeasure.instruments.rohdeschwarz.sfm.SFM $
pymeasure.instruments.yokogawa,214	property), 192
Monitor (class in pymeasure.display.listeners), 52	nicam_intercarrier_frequency (pymea-
Monitor (class in pymeasure.experiment.listeners), 42	sure.instruments.rohdeschwarz.sfm.SFM
motion_done (pymeasure.instruments.newport.esp300.Ax	
property), 168	nicam_IQ_inverted (pymea-
mouseMoved() (pymeasure.display.curves.Crosshairs	sure.instruments.rohdeschwarz.sfm.SFM
method), 49	property), 191
move() (pymeasure.instruments.anaheimautomation.DPSe method), 108	r <b>rieisdAnt_m6da1ypyline</b> asure.instruments.rohdeschwarz.sfm.SFM property), 192
move() (pymeasure.instruments.attocube.anc300.Axis	
method), 114	sure.instruments.rohdeschwarz.sfm.SFM
move() (pymeasure.instruments.parker.ParkerGV6	property), 192
method), 183	nicam_source (pymea-
	sure.instruments.rohdeschwarz.sfm.SFM
N	property), 192
<pre>new_curve() (pymeasure.display.widgets.ImageWidget</pre>	nicam_test_signal (pymea-
method), 55	sure.instruments.rohdeschwarz.sfm.SFM
<pre>new_curve() (pymeasure.display.widgets.PlotWidget</pre>	property), 192
method), 55	normal_channel (pymea-
new_curve() (pymeasure.display.widgets.TabWidget	sure.instruments.rohdeschwarz.sfm.SFM
method), 56	property), 192
next() (pymeasure.display.manager.ExperimentQueue method), 52	nxds (in module pymeasure.instruments.edwards), 120
next() (pymeasure.display.manager.Manager method),	0
53	offset (pymeasure.instruments.agilent.Agilent33220A
nicam_additional_bits (pymea-	property), 83
sure.instruments.rohdeschwarz.sfm.SFM	offset (pymeasure.instruments.agilent.Agilent33500
property), 191	property), 87
nicam_audio_frequency (pymea-	offset (pymeasure.instruments.agilent.agilent4156.VARD
sure.instruments.rohdeschwarz.sfm.SFM	property), 81
property), 191	offset (pymeasure.instruments.hp.HP33120A prop-
nicam_audio_volume (pymea-	erty), 122
sure.instruments.rohdeschwarz.sfm.SFM	offset_current (pymeasure.instruments.srs.SR570
property), 191	property), 205

offset_current_enabled (pymea-	$options \ (pymeasure. instruments. keysight. KeysightDSOX1102G$
sure.instruments.srs.SR570 property), 205	property), 161
offset_current_sign (pymea-	options (pymeasure.instruments.keysight.KeysightN5767A
sure.instruments.srs.SR570 property), 205	property), 163
offset_voltage (pymea-	output (pymeasure.instruments.agilent.Agilent33220A
sure.instruments.attocube.anc300.Axis prop-	property), 84
erty), 114	output (pymeasure.instruments.agilent.Agilent33500
open() (pymeasure.instruments.keithley.Keithley2750	property), 87
method), 157	output (pymeasure.instruments.anritsu.AnritsuMG3692C
open_all() (pymeasure.instruments.keithley.Keithley275	
method), 157 open_all_channels() (pymea-	output (pymeasure.instruments.srs.SR510 property), 204
open_all_channels() (pymea- sure.instruments.keithley.Keithley2700	output_1 (pymeasure.instruments.razorbill.razorbillRP100
method), 147	property), 185
open_channels (pymea-	output_2 (pymeasure.instruments.razorbill.razorbillRP100
sure.instruments.keithley.Keithley2700 prop-	property), 185
erty), 147	output_conversion() (pymea-
open_file_externally() (pymea-	sure.instruments.srs.SR830 method), 207
sure.display.windows.ManagedWindowBase	output_load(pymeasure.instruments.agilent.Agilent33500
method), 58	property), 87
open_rows_to_columns() (pymea-	output_low_grounded (pymea-
sure.instruments.keithley.Keithley2700	sure.instruments.keithley.Keithley6221 prop-
method), 147	erty), 150
operation_enable_reg (pymea-	output_off_state (pymea-
sure.instruments.rohdeschwarz.sfm.SFM	sure.instruments.keithley.Keithley2400 prop-
property), 192	erty), 137
operation_event_enabled (pymea-	output_trigger_on_external() (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keithley.Keithley2400
erty), 150	method), 137
operation_events (pymea-	output_trigger_on_external() (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keithley.Keithley6221
erty), 150	method), 151
options (pymeasure.instruments.bkprecision.BKPrecision	= :
property), 115	sure.instruments.attocube.anc300.Axis prop-
options (pymeasure.instruments.Instrument property), 62.	erty), 114 output_voltage (pymea-
options (pymeasure.instruments.keithley.Keithley2000	output_voltage (pymea- sure.instruments.rohdeschwarz.sfm.SFM
property), 130	property), 193
options (pymeasure.instruments.keithley.Keithley2260B	outputs_enabled (pymea-
property), 133	sure.instruments.ni.virtualbench.VirtualBench.PowerSupply
options (pymeasure.instruments.keithley.Keithley2400	property), 178
property), 137	1 1 7//
options (pymeasure.instruments.keithley.Keithley2450	P
property), 143	parallel_meas (pymea-
options (pymeasure.instruments.keithley.Keithley2600	sure.instruments.agilent.agilentB1500.AgilentB1500
property), 158	property), 94
options  (pymeasure. instruments. keithley. Keithley2700	Parameter (class in pymeasure.experiment.parameters),
property), 148	45
options (pymeasure.instruments.keithley.Keithley2750	parameter (pymeasure.display.inputs.Input property),
property), 157	51
options (pymeasure.instruments.keithley.Keithley6221	parameter_DAT1 (pymeasure.instruments.srs.SR860
property), 150	property), 211
options (pymeasure.instruments.keithley.Keithley6517B	parameter_DAT2 (pymeasure.instruments.srs.SR860
property), 155	property), 211

<pre>parameter_DAT3 (pymeasure.instruments.srs.SR860</pre>	phase (pymeasure.instruments.srs.SR830 property), 207 phase (pymeasure.instruments.srs.SR860 property), 211
parameter_DAT4 (pymeasure.instruments.srs.SR860	phase() (pymeasure.instruments.agilent.Agilent8722ES
property), 211	method), 69
parameter_objects() (pymea-	PhysicalParameter (class in pymea-
sure.experiment.procedure.Procedure method),	sure.experiment.parameters), 46
43	PLC (pymeasure.instruments.agilent.agilentB1500.ADCMode
<pre>parameter_values()</pre>	attribute), 102
sure.experiment.procedure.Procedure method), 43	plot() (pymeasure.experiment.experiment.Experiment method), 42
<pre>parameters_are_set()</pre>	<pre>plot_live() (pymeasure.experiment.experiment.Experimen</pre>
43	PlotFrame (class in pymeasure.display.widgets), 55
ParkerGV6 (class in pymeasure.instruments.parker), 183	Plotter (class in pymeasure.display.plotter), 53
parse() (pymeasure.experiment.results.Results method), 47	PlotterWindow (class in pymeasure.display.windows), 59
<pre>parse_axis() (pymeasure.display.widgets.PlotFrame</pre>	PlotWidget (class in pymeasure.display.widgets), 55 points (pymeasure.instruments.agilent.agilent4156.VAR2
<pre>parse_header() (pymeasure.experiment.results.Results</pre>	property), 81
static method), 47	polarity (pymeasure.instruments.danfysik.Danfysik8500
pattern_down (pymea-	property), 117
sure.instruments.attocube.anc300.Axis prop-	position (pymeasure.instruments.newport.esp300.Axis
erty), 114	property), 168
pattern_up (pymeasure.instruments.attocube.anc300.Axi	
property), 114	property), 183
pause() (pymeasure.instruments.agilent.agilentB1500.Ag	
method), 93	sure. instruments. parker. Parker GV6  property),
pause() (pymeasure.instruments.ami.AMI430 method),	183
106 pcolor() (pymeasure.experiment.experiment.Experiment	power (pymeasure.instruments.agilent.Agilent8257D property), 68
method), 42	power (pymeasure.instruments.anapico.APSIN12G prop-
peak_search(pymeasure.instruments.anritsu.AnritsuMS9	
property), 111	power (pymeasure.instruments.anritsu.AnritsuMG3692C
period (pymeasure.instruments.keithley.Keithley2000	property), 110
property), 130	power (pymeasure.instruments.keithley.Keithley2260B
period_aperature (pymea-	property), 133
sure.instruments.keithley.Keithley2000 prop-	power (pymeasure.instruments.thorlabs.ThorlabsPM100USB
erty), 130	property), 214
period_digits (pymea-	preemphasis_enabled (pymea-
sure.instruments.keithley.Keithley2000 property), 130	sure.instruments.rohdeschwarz.sfm.Sound_Channel property), 196
period_reference (pymea-	preemphasis_time (pymea-
sure.instruments.keithley.Keithley2000 prop-	sure.instruments.rohdeschwarz.sfm.Sound_Channel
erty), 130	property), 196
period_threshold (pymea-	prepare() (pymeasure.display.curves.BufferCurve
sure.instruments.keithley.Keithley2000 prop-	method), 49
erty), 130	Procedure (class in pymeasure.experiment.procedure),
phase (pymeasure.instruments.agilent.Agilent33500	43
property), 87	program_sweep() (pymea-
phase (pymeasure.instruments.ametek.Ametek7270 prop-	sure.instruments.oxfordinstruments.ITC503
	method), 181
erty), 104 phase (pymeasure.instruments.signalrecovery.DSP7265	PrologixAdapter (class in pymeasure.adapters), 34
property), 202	
phase (pymeasure.instruments.srs.SR510 property), 204	sure.instruments.agilent.Agilent33220A

property), 84	pymeasure.experiment.listeners
pulse_dutycycle (pymea-	module, 42
sure.instruments.agilent.Agilent33500 property), 87	<pre>pymeasure.experiment.parameters   module, 44</pre>
pulse_frequency (pymea-	pymeasure.experiment.procedure
sure.instruments.agilent.Agilent8257D prop-	module, 43
erty), 68	pymeasure.experiment.results
pulse_hold(pymeasure.instruments.agilent.Agilent33220	
property), 84	pymeasure.experiment.workers
pulse_hold(pymeasure.instruments.agilent.Agilent33500	
property), 87	pymeasure.instruments
pulse_input (pymeasure.instruments.agilent.Agilent8257	D module, 59
property), 68	pymeasure.instruments.advantest
pulse_period (pymea-	module, 65
sure.instruments.agilent.Agilent33220A	pymeasure.instruments.advantest.advantestR3767CG
property), 84	module, 66
pulse_period (pymea-	pymeasure.instruments.agilent
sure.instruments.agilent.Agilent33500 prop-	module, 66
erty), 87	pymeasure.instruments.agilent.agilent4156
pulse_source (pymea-	module, 76
sure.instruments.agilent.Agilent8257D prop-	pymeasure.instruments.agilent.agilentB1500
erty), 68	module, 101
pulse_transition (pymea-	pymeasure.instruments.ametek
sure.instruments.agilent.Agilent33220A	module, 103
property), 84	pymeasure.instruments.ami
pulse_transition (pymea-	module, 105
sure.instruments.agilent.Agilent33500 prop-	pymeasure.instruments.anaheimautomation
erty), 87	module, 107
pulse_width(pymeasure.instruments.agilent.Agilent3322	
property), 84	module, 109
pulse_width (pymeasure.instruments.agilent.Agilent3350 property), 87	module, 109
pymeasure.display.browser	pymeasure.instruments.attocube
module, 49	module, 112
pymeasure.display.curves	pymeasure.instruments.bkprecision
module, 49	module, 115
pymeasure.display.inputs	pymeasure.instruments.comedi
module, 50	module, 65
pymeasure.display.listeners	pymeasure.instruments.danfysik
module, 52	module, 116
pymeasure.display.log	pymeasure.instruments.deltaelektronika
module, 52	module, 119
pymeasure.display.manager	pymeasure.instruments.edwards
module, 52	module, 120
pymeasure.display.plotter	pymeasure.instruments.fluke
module, 53	module, 120
pymeasure.display.thread	<pre>pymeasure.instruments.fwbell</pre>
module, 54	module, 120
pymeasure.display.widgets	<pre>pymeasure.instruments.hp</pre>
module, 54	module, 122
pymeasure.display.windows	<pre>pymeasure.instruments.keithley</pre>
module, 57	module, 126
pymeasure.experiment.experiment	pymeasure.instruments.keysight
module, 41	module, 158

pymeasure.instruments.lakeshore	method), 173	
module, 164	query_arbitrary_waveform_gain_a	
pymeasure.instruments.newport	* *	tualbench. Virtual Bench. Function General Bench and General Ben
module, 167	method), 173	
pymeasure.instruments.ni	<pre>query_current_output()</pre>	(pymea-
module, 168		ch.VirtualBench.PowerSupply
pymeasure.instruments.oxfordinstruments	method), 178	
module, 180	<pre>query_dc_current()</pre>	(рутеа-
pymeasure.instruments.parker module,182	sure.instruments.ni.virtualben method), 171	ch.VirtualBench.DigitalMultimeter
pymeasure.instruments.pendulum	<pre>query_dc_voltage()</pre>	(pymea-
module, 184	sure.instruments.ni.virtualben	ch.VirtualBench.DigitalMultimeter
pymeasure.instruments.razorbill	method), 171	
module, 185	query_enabled_analog_channels()	) (pymea-
pymeasure.instruments.rohdeschwarz		ch.VirtualBench.MixedSignalOscill
module, 186	method), 176	Ŭ
pymeasure.instruments.signalrecovery	<pre>query_export_signal()</pre>	(pymea-
module, 200		ch.VirtualBench.DigitalInputOutpu
pymeasure.instruments.srs	method), 170	
module, 203	query_generation_status()	(рутеа-
pymeasure.instruments.tektronix		ch.VirtualBench.FunctionGenerato
module, 213	<i>method</i> ), 173	
pymeasure.instruments.thorlabs	query_learn()	(рутеа-
module, 213	sure.instruments.agilent.agile	
pymeasure.instruments.validators	method), 93	212 0011.3.11.11.212 00
module, 63	query_learn()	(рутеа-
pymeasure.instruments.yokogawa	sure.instruments.agilent.agile	4.5
module, 214	static method), 99	mb15 00.Quel y2ealli
module, 211	query_learn()	(рутеа-
Q	sure.instruments.agilent.agile	4.5
	method), 97	mB1300.5M0
QListener (class in pymeasure.display.listeners), 52		(рутеа-
query_ac_current() (pyme		
sure.instruments.ni.virtualbench.VirtualBench method), 171	method), 93	
query_acquisition_status() (pyme	a- query_learn_header()	(рутеа-
sure.instruments.ni.virtualbench.VirtualBench method), 176	class method), 100	
query_adc_setup() (pyme	$a$ - query_line_configuration()	(pymea-
sure.instruments.agilent.agilentB1500.Agilen method), 95		ch.VirtualBench.DigitalInputOutpu
	a- query_meas_mode()	(pymea-
sure.instruments.ni.virtualbench.VirtualBench method), 176	h.MixedSignal <b>OstAlivstropp</b> ents.agilent.agile. method), 94	ntB1500.AgilentB1500
query_analog_channel_characteristics()	<pre>query_meas_op_mode()</pre>	(pymea-
(pymeasure.instruments.ni.virtualbench.Virtu method), 176	alBench.Mixed <b>SignillOkuMentsops</b> ilent.agile. method), 97	ntB1500.AgilentB1500
	$a_{a}$ query_meas_range_current_auto()	) (pymea-
sure. instruments. ni. virtual bench. Virtual Bench in the property of the p	h.MixedSignal <b>Outqlipstropp</b> ents.agilent.agile	
method), 176	method), 97	(
<pre>query_analog_pulse_width_trigger() (pyme</pre>	a_ query_meas_ranges()	(pymea- ntP1500 AcilentP1500
sure.instruments.ni.virtualbench.VirtualBench method), 176	method), 97	
query_arbitrary_waveform() (pyme	a- query_meas_settings()	(рутеа-
sure.instruments.ni.virtualbench.VirtualBench	h.FunctionGen <b>ewae.i</b> nstruments.agilent.agile.	ntB1500.AgilentB1500

method), 94		queue()	(py measure. display. manager. Manager	method),
query_measurement()	(рутеа-		53	
	alBench.Di	gitudikkildi	i <b>hpestm</b> easure.display.windows.Managed	WindowBase
method), 171			method), 58	
query_modules()	4 .	-	equence()	(pymea-
sure.instruments.agilent.agilentB1500	.AgilentB1.	500	sure.display.widgets.SequencerWidget	method),
method), 93			56	~~~~
query_sampling_settings()	(pymea-		= :	rs.SR830
sure.instruments.agilent.agilentB1500.	.AgilentB1.	500	method), 207	
method), 96	,	R		
<pre>query_series_resistor()</pre>	(pymea-			
sure.instruments.agilent.agilentB1500.	.AgilentB1.	<sup>0</sup> 875_out	(pymeasure.instruments.rohdeschwarz.	sfm.SFM
method), 96	(		property), 186	
<pre>query_staircase_sweep_settings()</pre>	(pymea-		(pymeasure.instruments.ami.AMI430	method),
sure.instruments.agilent.agilentB1500.	.Адиепты 1.		106	
method), 95	(	ramp_ra	te_current	(pymea-
query_standard_waveform()	(pymea-	un ati an Car		roperty),
sure.instruments.ni.virtualbench.Virtua	швепсп.ғ		100	
<pre>method), 173 query_time_stamp_setting()</pre>	(рутеа-	ramp_ra	te_field (pymeasure.instruments.ami	.AM1430
sure.instruments.agilent.agilentB1500.		500	property), 106	
method), 95	.Адиешь1.	ramp_so		(pymea-
query_timing()	(рутеа-		sure.instruments.agilent.agilentB1500.	SMU
sure.instruments.ni.virtualbench.Virtua	A 2	ivadSiana	method), 98  Oscilloscope	-
method), 176	ш Бенен.т	ramps/sy		(pymea-
query_trigger_delay()	(рутеа-		sure.instruments.agilent.Agilent33220	4
sure.instruments.ni.virtualbench.Virtua		ixedSiona	property), 84 lOscilloscope	(mymag
method), 177		·· rainp <u>s</u> ·sy	sure.instruments.agilent.Agilent33500	(pymea- prop-
query_trigger_type()	(рутеа-		erty), 87	prop-
sure.instruments.ni.virtualbench.Virtua		ixedSiona	lOscilloscope	(рутеа-
method), 177		1-ainp <u>o-</u> co	sure.instruments.ami.AMI430 method)	
query_voltage_output()	(рутеа-	ramn to	o_current()	, 100 (pymea-
		werSuppl	y sure.instruments.danfysik.Danfysik850	(pymea 10
method), 178		11.	method), 117	·
query_waveform_mode()	(рутеа-	ramp to	current()	(рутеа-
sure.instruments.ni.virtualbench.Virtua	alBench.Fu	ınctionGei	างเสียงโกรtruments.deltaelektronika.SM7	045D
method), 173			method), 119	
QueryLearn (class in	рутеа-	ramp_to	_current()	(рутеа-
sure.instruments.agilent.agilentB1500	), 99	• -	sure.instruments.keithley.Keithley2400	
questionable_event_enabled	(pymea-		method), 137	
sure.instruments.keithley.Keithley6221	prop-	ramp_to	_current()	(pymea-
erty), 151		_	sure.instruments.keithley.Keithley2450	
questionable_event_reg	(рутеа-		method), 143	
sure.instruments.rohdeschwarz.sfm.SF	M	ramp_to	_current()	(рутеа-
property), 193			sure.instruments.yokogawa.Yokogawa?	7651
questionable_events	(рутеа-		method), 215	
sure.instruments.keithley.Keithley6221	prop-	ramp_to	_field() (pymeasure.instruments.ami	.AMI430
erty), 151			method), 106	
questionable_operation_enable_reg	(рутеа-	ramp_to	_voltage()	(pymea-
sure.instruments.rohdeschwarz.sfm.SF	'M		sure.instruments.keithley.Keithley2400	)
property), 193	,		method), 137	
questionanble_status_reg	(рутеа-	ramp_to	_voltage()	(pymea-
sure.instruments.rohdeschwarz.sfm.SF	M		sure.instruments.keithley.Keithley2450	1
property), 193			method), 143	

ramp_to	o_voltage() (pymea- sure.instruments.keithley.Keithley6517B method), 155	read_buffer() sure.instr method),	ruments.pendulum.cnt91.CNT91	vmea-
ramp_to	o_voltage() (pymea- sure.instruments.yokogawa.Yokogawa7651	read_bytes()  method),	(pymeasure.adapters.VISAAa	lapter
	method), 216	read_channels()	4.5	утеа-
ramp_to	o_zero() (pymea-	sure.inst	ruments.agilent.agilentB1500.Agi	ilentB1500
	sure.instruments.deltaelektronika.SM7045D	method),	96	
	method), 119	read_data()(pyn	neasure.instruments.agilent.agile	ntB1500.AgilentB1500
range (	pymeasure.instruments.fwbell.FWBell5080 prop-	method),	96	
	erty), 121	read_memory()	(p)	утеа-
range	(pymeasure.instruments.hp.HP3478A property),	sure.inst	ruments.anritsu.AnritsuMS9710C	7
	125	method),	111	
range	(pymeasure.instruments.lakeshore.LakeShore425	read_output()	(p)	vmea-
_	property), 167	sure.inst	ruments.ni.virtualbench.VirtualBe	ench.PowerSupply
Ranging		method),		11 2
5 .	sure.instruments.agilent.agilentB1500), 100	read_raw()	(pymeasure.adapters.VXI11Aa	lapter
ratio(r	pymeasure.instruments.agilent.agilent4156.VARD	method),		<sub>F</sub>
	property), 81	read_trace()		утеа-
ratio (	(pymeasure.instruments.signalrecovery.DSP7265		ruments.rohdeschwarz.fsl.FSL	, meed
14010	property), 202	method),		
razorbi	illRP100 (class in pymea-		ıle pymeasure.instruments.comed	li) 65
1 0201 01	sure.instruments.razorbill), 185		= -	
maad()			measure.experiment.listeners.Lis	sierier
	(pymeasure.adapters.Adapter method), 31	method),		4
	(pymeasure.adapters.FakeAdapter method), 32		re.display.log.LogHandler.Emitte	er at-
read()	(pymeasure.adapters.PrologixAdapter method),	tribute),		12
10	35		n pymeasure.experiment.listeners	
	(pymeasure.adapters.SerialAdapter method), 34		isure.instruments.signalrecovery.	DSP/203
	(pymeasure.adapters.TelnetAdapter method), 40	property		
	(pymeasure.adapters.VISAAdapter method), 37	reference_exte		утеа-
	(pymeasure.adapters.VXI11Adapter method), 39		uments.srs.SR860 property), 211	
read()	(py measure. instruments. attocube. adapters. Attocube			утеа-
	method), 113		ruments.anapico.APSIN12G prop	erty),
read()	(pymeasure.instruments.dan fysik.Dan fysik Adapter	109		
	method), 116	reference_phase		утеа-
read()	(py measure. instruments. fwbell. FWB ell 5080	sure.inst	uments.signalrecovery.DSP7265	
	method), 121	property		
read()	(pymeasure.instruments.Instrument method), 62	reference_sour	ce (pymeasure.instruments.srs.S	SR830
read()	(pymeasure.instruments.keysight.KeysightDSOX11)	02G property	, 207	
	method), 161	reference_sour	ce (pymeasure.instruments.srs.S	SR860
read()	(pymeasure.instruments.lakeshore.LakeShoreUSB.	Adapter property	, 211	
	method), 164	reference_trig	germode (py	утеа-
read()	(pymeasure.instruments.ni.virtualbench.VirtualBe			
**	method), 170	refresh_parame		утеа-
read()	(pymeasure.instruments.ni.virtualbench.VirtualBe			
	method), 171	43	, <b>,</b>	,,
read()	(pymeasure.instruments.parker.ParkerGV6		(pymeasure.experiment.results.R	esults
	method), 183	method),		
read ar			ure.instruments.danfysik.Danfysi	:k8500
_ cau_ui	sure.instruments.ni.virtualbench.VirtualBench.M			
	method), 177		sure.instruments.keithley.Keithley	v2000
read ar	nalog_digital_u64() (pymea-	method),		y2000
- cau_di	sure.instruments.ni.virtualbench.VirtualBench.M			vmea-
	method), 177	~	орв uments.rohdeschwarz.sfm.SFM	ymcu-
	memou), 1//	sure.iilsti	umenus.ronaesenwarz.sjm.sr/W	

property), 193	sure.instruments.keithley.Keithley2000
remote_local_state (pymea-	method), 130
sure.instruments.agilent.Agilent33220A	reset_buffer() (pymea-
property), 84	sure.instruments.keithley.Keithley2400
remote_lock() (pymea-	method), 138
sure.instruments.keithley.Keithley2000	reset_buffer() (pymea-
method), 130	sure.instruments.keithley.Keithley2450
remove() (pymeasure.display.manager.Manager	method), 144
method), 53	reset_buffer() (pymea-
remove() (pymeasure.display.widgets.ImageWidget	sure.instruments.keithley.Keithley2700
method), 55	method), 148
remove() (pymeasure.display.widgets.PlotWidget	reset_buffer() (pymea-
method), 55	sure.instruments.keithley.Keithley6221
remove() (pymeasure.display.widgets.TabWidget	method), 151
method), 56	reset_buffer() (pymea-
replace_placeholders() (in module pymea-	sure.instruments.keithley.Keithley6517B
sure.experiment.results), 47	method), 155
res_bandwidth (pymea-	reset_instrument() (pymea-
sure.instruments.rohdeschwarz.fsl.FSL prop-	sure. instruments. ni. virtual bench. Virtual Bench. Digital Input Output the properties of the prop
erty), 200	method), 170
$\verb"reset()" (pymeasure.instruments.agilent.agilentB1500.AgilentB1500.$	il <b>resBet5<u>0</u>0</b> nstrument() (pymea-
method), 93	sure. instruments. ni. virtual bench. Virtual Bench. Digital Multimeter
reset() (pymeasure.instruments.bkprecision.BKPrecision	19130B method), 171
method), 115	reset_instrument() (pymea-
reset() (pymeasure.instruments.fwbell.FWBell5080	sure. instruments. ni. virtual bench. Virtual Bench. Function Generato
method), 121	method), 173
reset() (pymeasure.instruments.hp.HP3478A method),	reset_instrument() (pymea-
126	sure. instruments. ni. virtual bench. Virtual Bench. Mixed Signal Oscillari, and the property of the propert
reset() (pymeasure.instruments.Instrument method), 62	method), 177
reset() (pymeasure.instruments.keithley.Keithley2000	reset_instrument() (pymea-
method), 130	sure. instruments. ni. virtual bench. Virtual Bench. Power Supply
reset() (pymeasure.instruments.keithley.Keithley2260B	method), 178
method), 133	reset_interlocks() (pymea-
reset() (pymeasure.instruments.keithley.Keithley2400	sure.instruments.danfysik.Danfysik8500
method), 138	method), 117
reset() (pymeasure.instruments.keithley.Keithley2450	
method), 143	sure. instruments. an ahe imautomation. DPS eries Motor Controller
reset() (pymeasure.instruments.keithley.Keithley2600	method), 108
method), 158	<pre>reset_time() (pymeasure.instruments.Mock method),</pre>
reset() (pymeasure.instruments.keithley.Keithley2700	63
method), 148	resistance(pymeasure.instruments.agilent.Agilent34410A
reset() (pymeasure.instruments.keithley.Keithley2750	property), 72
method), 157	resistance(pymeasure.instruments.agilent.Agilent34450A
reset() (pymeasure.instruments.keithley.Keithley6221	property), 75
method), 151	resistance (pymeasure.instruments.hp.HP34401A
reset() (pymeasure.instruments.keithley.Keithley6517B	property), 123
method), 155	resistance (pymeasure.instruments.keithley.Keithley2000
${\tt reset()}\ (pymeasure.instruments.keysight.KeysightDSOXI)$	
method), 161	resistance (pymeasure.instruments.keithley.Keithley2400
reset() (pymeasure.instruments.keysight.KeysightN5767A	
method), 163	resistance (pymeasure.instruments.keithley.Keithley2450
reset() (pymeasure.instruments.parker.ParkerGV6	property), 144
method), 183	resistance (pymeasure.instruments.keithley.Keithley6517B
reset_buffer() (pymea-	property), 156

resistance_4w	(pymea-	sure.instruments.keithley.Keithley2400	) prop-
sure.instruments.agilent.Agilent34-	410A	erty), 138	
property), 72		resistance_range	(pymea-
resistance_4w	(pymea-	sure.instruments.keithley.Keithley2450	) prop-
sure.instruments.agilent.Agilent34-	450A	erty), 144	
property), 75		resistance_range	(pymea-
resistance_4w (pymeasure.instruments.h	p.HP34401A	sure.instruments.keithley.Keithley6517	7B
property), 123		property), 156	
resistance_4w_auto_range		resistance_reference	(pymea-
sure.instruments.agilent.Agilent34- property), 75	450A	sure.instruments.keithley.Keithley2000 erty), 131	
resistance_4W_digits	(pymea-	resistance_resolution	(рутеа-
sure.instruments.keithley.Keithley 2	000 prop-	sure.instruments.agilent.Agilent34450	A
erty), 130		property), 75	
resistance_4W_nplc	(pymea-	resolution (pymeasure.instruments.anritsu.An	iritsuMS9710C
sure.instruments.keithley.Keithley2	000 prop-	property), 111	
erty), 130		resolution (pymeasure.instruments.hp.HP347	78A prop-
resistance_4w_range	(рутеа-	erty), 126	
sure.instruments.agilent.Agilent34-	450A	resolution_actual	(рутеа-
property), 75		sure.instruments.anritsu.AnritsuMS97	10C
resistance_4W_range	(pymea-	property), 111	
sure.instruments.keithley.Keithley2	000 prop-	resolution_vbw	(pymea-
erty), 130	,	sure.instruments.anritsu.AnritsuMS97	10C
resistance_4W_reference	(pymea-	property), 111	
sure.instruments.keithley.Keithley2	000 prop-	Results (class in pymeasure.experiment.results	
erty), 130	(	ResultsClass (pymeasure.display.widgets.Ima	ageFrame
resistance_4w_resolution	(pymea-	attribute), 55	N T
sure.instruments.agilent.Agilent34- property), 75	450A	ResultsClass (pymeasure.display.widgets.Pattribute), 55	
resistance_auto_range	(pymea-	ResultsCurve (class in pymeasure.display.cur	
sure.instruments.agilent.Agilent34-	450A	ResultsDialog (class in pymeasure.display.wid	
property), 75	,	ResultsImage (class in pymeasure.display.cur	
resistance_digits	(pymea-	resume() (pymeasure.display.manager.	Manager
sure.instruments.keithley.Keithley2	000 prop-	method), 53	
erty), 130	,	rf_out_enabled	(pymea-
resistance_nplc	(pymea-	sure.instruments.rohdeschwarz.sfm.SF	<sup>r</sup> M
sure.instruments.keithley.Keithley2	000 prop-	property), 193	
erty), 131	(		(pymea-
resistance_nplc	(pymea-	sure.instruments.rohdeschwarz.sfm.SF	MI.
sure.instruments.keithley.Keithley2	400 prop-	property), 193	(
erty), 138	(	rf_sweep_span	(pymea-
resistance_nplc	(pymea-	sure.instruments.rohdeschwarz.sfm.SF	IVI
sure.instruments.keithley.Keithley2	450 prop-	property), 193 rf_sweep_start	(mymag
erty), 144 resistance_nplc	(рутеа-	sure.instruments.rohdeschwarz.sfm.SF	(рутеа- гм
sure.instruments.keithley.Keithley6	* *	property), 193	IVI
property), 156	317 <b>B</b>	rf_sweep_step	(рутеа-
resistance_range	(рутеа-	sure.instruments.rohdeschwarz.sfm.SF	4.2
sure.instruments.agilent.Agilent34-	* *	property), 193	IVI
property), 75	r50/1	rf_sweep_stop	(рутеа-
resistance_range	(рутеа-	sure.instruments.rohdeschwarz.sfm.SF	4.0
sure.instruments.keithley.Keithley2	* *	property), 193	174
erty), 131	ooo prop	right_limit (pymeasure.instruments.newport.	esp300.Axis
resistance range	(pymea-	property). 168	r

round_up() (pymeasure.display.curves.ResultsImage method), 50	scan_single() (pymea- sure.instruments.agilent.Agilent8722ES
rsd (pymeasure.instruments.deltaelektronika.SM7045D property), 120	method), 70 ScientificInput (class in pymeasure.display.inputs),
run() (pymeasure.display.plotter.Plotter method), 53	51
run() (pymeasure.experiment.workers.Worker method), 46	screen_layout (pymeasure.instruments.srs.SR860 property), 211
run() (pymeasure.instruments.keysight.KeysightDSOX110 method), 161	
run() (pymeasure.instruments.ni.virtualbench.VirtualBen	cl <b>sElufctialGbrarta(i)</b> (pymea- sure.instruments.ni.virtualbench.VirtualBench.FunctionGenerato
method), 173 run() (pymeasure.instruments.ni.virtualbench.VirtualBen	
method), 177	send_trigger() (pymea-
memou), 177	sure.instruments.agilent.agilentB1500.AgilentB1500
S	method), 94
sample_continuously() (pymea-	sensitivity (pymeasure.instruments.ametek.Ametek7270
sure.instruments.keithley.Keithley2400	property), 104
method), 138	sensitivity (pymeasure.instruments.signalrecovery.DSP7265 property), 202
sample_frequency (pymeasure.instruments.srs.SR830	sensitivity (pymeasure.instruments.srs.SR510 prop-
property), 207 SAMPLING (pymeasure.instruments.agilent.agilentB1500.M	
attribute), 102	sensitivity (pymeasure.instruments.srs.SR570 prop-
sampling_auto_abort() (pymea-	erty), 205
sure.instruments.agilent.agilentB1500.AgilentB1	580nsitivity (pymeasure.instruments.srs.SR830 prop-
method), 96	erty), 207
sampling_mode (pymea-	sensitvity (pymeasure.instruments.srs.SR860 prop-
sure.instruments.agilent.agilentB1500.AgilentB1	9500 erty), 211
property), 96	SequenceEvaluationException, 56
sampling_points (pymea- sure.instruments.anritsu.AnritsuMS9710C	SequencerWidget (class in pymeasure.display.widgets), 56
property), 111	$\verb serial_baud  (pymeasure. instruments. rohdeschwarz. sfm. SFM $
sampling_source() (pymea-	property), 194
sure.instruments.agilent.agilentB1500.SMU method), 99	serial_bits(pymeasure.instruments.rohdeschwarz.sfm.SFM property), 194
sampling_timing() (pymea-	serial_flowcontrol (pymea-
sure.instruments.agilent.agilentB1500.AgilentB1	sure.instruments.rohdeschwarz.sfm.SFM property), 194
method), 96 SamplingMode (class in pymea-	serial_nr (pymeasure.instruments.attocube.anc300.Axis
SamplingMode (class in pymea- sure.instruments.agilent.agilentB1500), 102	property), 114
SamplingPostOutput (class in pymea-	serial_parity (pymea-
sure.instruments.agilent.agilentB1500), 103	sure.instruments.rohdeschwarz.sfm.SFM
${\sf save}$ ( ) (pymeasure.instruments.agilent.agilent4156.Agilent ( ) (	nt4156 property), 194
method), 79	serial_stopbits (pymea-
save_var() (pymeasure.instruments.agilent.agilent4156 method), 79	property), 194
scale_volt(pymeasure.instruments.rohdeschwarz.sfm.Si	FigerialAdapter (class in pymeasure.adapters), 33
property), 194	series_resistance (pymea-
scan() (pymeasure.instruments.agilent.Agilent8722ES	sure.instruments.agilent.agilent4156.SMU
method), 69	property), 80
scan_continuous() (pymea-	series_resistor (pymea-
sure.instruments.agilent.Agilent8722ES method), 69	sure.instruments.agilent.agilentB1500.SMU property), 97
meinoa), 69 scan_points(pymeasure.instruments.agilent.Agilent8722	
property), 70	sure.instruments.agilent.Agilent8722ES
F. CF C. 2777, C	

method), 70	sure.instruments.danfysik.Danfysik8500
set_buffer() (pyme	
sure.instruments.signalrecovery.DSP7265	set_ramp_to_current() (pymea-
method), 202	sure.instruments.danfysik.Danfysik8500
<pre>set_channel_A_mode()</pre> <pre>(pyme</pre>	
sure.instruments.ametek.Ametek7270 method	(), set_scaling() (pymeasure.instruments.srs.SR830
104	method), 207
$set\_color()$ (pymeasure.display.widgets.PlotWidgets.	
method), 56	sure.instruments.danfysik.Danfysik8500
<pre>set_color() (pymeasure.display.widgets.TabWidg</pre>	
method), 56	set_software_limits() (pymea-
set_defaults() (pymeasure.adapters.PrologixAdapt	
method), 36	183
set_defaults() (pyme	
sure.instruments.parker.ParkerGV6 method	•
183	sure.instruments.keithley.Keithley2400
set_differential_mode() (pyme	
sure.instruments.ametek.Ametek7270 method	•
104	sure.instruments.keithley.Keithley6221
set_fixed_frequency() (pyme	
sure.instruments.agilent.Agilent8722ES	
method), 70	sure.instruments.keithley.Keithley2400
set_hardware_limits() (pyme	
sure.instruments.parker.ParkerGV6 method 183	
	sure.instruments.ametek.Ametek7270 method), a- 104
set_IF_bandwidth() (pyme sure.instruments.agilent.Agilent8722ES	
method), 70	
	sure.instruments.signalrecovery.DSP7265 ck method), 202
<pre>set_output_voltage() (pymeasure.instruments.Mo     method), 63</pre>	setpoint_1(pymeasure.instruments.lakeshore.LakeShore331
set_parameter() (pyme	
sure.display.inputs.BooleanInput method	
50	property), 165
	a- setting() (pymeasure.instruments.Instrument static
sure.display.inputs.FloatInput method), 50	method), 62
	ut setting() (pymeasure.instruments.keysight.KeysightDSOX1102G
method), 51	static method), 161
	a- setup_plot() (pymeasure.display.plotter.Plotter
sure.display.inputs.IntegerInput method	
51	SFM (class in pymeasure.instruments.rohdeschwarz.sfm),
<pre>set_parameter() (pymeasure.display.inputs.ListInp</pre>	
method), 51	shape (pymeasure.instruments.agilent.Agilent33220A
set_parameter() (pyme	
sure.display.inputs.ScientificInput method	
51	property), 87
<pre>set_parameters()</pre>	
sure.display.windows.ManagedWindowBase	122
method), 59	<pre>shutdown() (pymeasure.experiment.procedure.Procedure</pre>
set_parameters() (pyme	
sure.experiment.procedure.Procedure method	
43	method), 47
set_point (pymeasure.instruments.fluke.Fluke73-	shutdown() (pymeasure.instruments.agilent.Agilent8257D
property), 120	method), 68
set ramp delay() (nyme	a- shutdown() (nymeasure instruments ametek Ametek 7270

method),	104		property), 205	
shutdown()	(pymeasure.instruments.ami.AMI430	sine_am	plitudepreset1	(pymea-
method),	106		sure.instruments.srs.SR860 property),	212
shutdown()(pyme	easure.instruments.anritsu.AnritsuMG36	<b>9216</b> 0e_amj	plitudepreset2	(pymea-
method),	110		$sure. instruments. srs. SR 860\ property),$	212
shutdown()(pyme	easure.instruments.bkprecision.BKPrecis	s iso intoled <u>3</u> 60nB <sub>1</sub>	plitudepreset3	(pymea-
method),	115		$sure. instruments. srs. SR 860\ property),$	212
shutdown()(pyme	easure.instruments.deltaelektronika.SM7	<b>%151@</b> _amj	plitudepreset4	(pymea-
method),	120		$sure. instruments. srs. SR 860\ property),$	212
<pre>shutdown()</pre>	(pymeasure.instruments.hp.HP3478A	sine_dc	levelpreset1	(pymea-
method),	126		$sure. instruments. srs. SR 860\ property),$	212
<pre>shutdown()</pre>	(pymeasure.instruments.Instrument			(рутеа-
method),			$sure. instruments. srs. SR 860\ property),$	212
shutdown()(pyme	easure.instruments.keithley.Keithley2000			(pymea-
method),			$sure. instruments. srs. SR 860\ property),$	212
	easure.instruments.keithley.Keithley2260			(pymea-
method),			$sure. instruments. srs. SR 860\ property),$	
shutdown() (pyme method),	easure.instruments.keithley.Keithley2400 138		ltage (pymeasure.instruments.srs.SR8 erty), 207	330 prop-
shutdown()(pyme	easure.instruments.keithley.Keithley2450	sine_vo	ltage (pymeasure.instruments.srs.SR&	360 prop-
method),	144		erty), 212	
shutdown()(pyme	easure.instruments.keithley.Keithley2600	single(	) (pymeasure.instruments.keysight.Key.	sightDSOX1102G
method),	158		method), 161	
shutdown()(pyme	easure.instruments.keithley.Keithley2700	single_	sweep()	(pymea-
method),	148		sure.instruments.anritsu.AnritsuMS97	10C
shutdown()(pyme	easure.instruments.keithley.Keithley2750	)	method), 111	
method),	157	single_	sweep()	(pymea-
<pre>shutdown() (pyme</pre>	easure.instruments.keithley.Keithley6221	1	sure.instruments.rohdeschwarz.fsl.FSL	
method),	151		method), 200	
shutdown()(pyme	easure.instruments.keithley.Keithley6517	<b>B</b> lew_ra	${\sf te}(py measure. instruments. dan fy sik. Data and the contractions of the contraction of the contracti$	infysik8500
method),	156		property), 118	
<pre>shutdown() (pyme</pre>	easure.instruments.keysight.KeysightDSC	<i>O\$\]Id\@2f</i> a	${\sf te\_1}$ (pymeasure.instruments.razorbill.	razorbillRP100
method),			property), 185	
shutdown()(pyme	easure.instruments.keysight.KeysightN57	<b>%7.4e</b> w_ra	${\sf te\_2}$ (pymeasure.instruments.razorbill.	razorbillRP100
method),			property), 185	
	measure.instruments.newport.ESP300			270 prop-
method),			erty), 104	
	easure.instruments.ni.virtualbench.Virtu			DSP7265
method),			property), 203	
	easure.instruments.ni.virtualbench.Virtu			sPro8000
method),			property), 214	
	easure.instruments.ni.virtualbench.Virtu			рутеа-
method),			sure.instruments.deltaelektronika), 119	9
	easure.instruments.ni.virtualbench.Virtu			рутеа-
method),			sure.instruments.agilent.agilent4156),	79
	easure.instruments.ni.virtualbench.Virtu		· ·	pymea-
method),			sure.instruments.agilent.agilentB1500	
	easure.instruments.ni.virtualbench.Virtu		11 7	(pymea-
method),			sure.instruments.agilent.agilentB1500	.wait1ime1ype
	easure.instruments.signalrecovery.DSP7		attribute), 103	on +D1500 A - H - 4D1500
method),			es (pymeasure.instruments.agilent.agil	emB1300.AguentB1300
method),	easure.instruments.yokogawa.Yokogawa? 216	smu_ref	property), 93	(mymag
				(pymea-
signar_inverted	d (pymeasure.instruments.srs.SR570		sure.instruments.agilent.agilentB1500	Aguemb1500

property), 93	erty), 151
SMU_SOURCE (pymeasure.instruments.agilent.agilentB1500	
attribute), 103	sure.instruments.keithley.Keithley2400 prop-
SMUCurrentRanging (class in pymea-	
sure.instruments.agilent.agilentB1500), 100	source_enabled (pymea-
SMUVoltageRanging (class in pymea-	
sure.instruments.agilent.agilentB1500), 101	property), 115
snap() (pymeasure.instruments.srs.SR830 method), 208	
snap() (pymeasure.instruments.srs.SR860 method), 212	sure.instruments.keithley.Keithley2400 prop-
Sound_Channel (class in pymea-	erty), 138
sure.instruments.rohdeschwarz.sfm), 196	source_enabled (pymea-
${\tt sound\_mode} \ (py measure. instruments. rohdeschwarz. sfm. SF$	FM sure.instruments.keithley.Keithley2450 prop-
property), 194	erty), 144
source_auto_range (pymea-	source_enabled (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keithley.Keithley6221 prop-
erty), 151	erty), 151
source_compliance (pymea-	source_enabled (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.keithley.Keithley6517B
erty), 151	property), 156
source_current (pymea-	source_enabled (pymea-
sure.instruments.keithley.Keithley2400 prop-	sure.instruments.yokogawa.Yokogawa7651
erty), 138	property), 216
source_current (pymea-	source_enabled (pymea-
sure.instruments.keithley.Keithley2450 prop-	sure.instruments.yokogawa.YokogawaGS200
erty), 144	property), 216
source_current (pymea-	source_level (pymea-
sure.instruments.keithley.Keithley6221 prop-	sure.instruments.yokogawa.YokogawaGS200
erty), 151	property), 216
source_current (pymea-	
sure.instruments.yokogawa.Yokogawa7651	property), 138
property), 216	source_mode (pymeasure.instruments.keithley.Keithley2450
source_current_delay (pymea-	property), 144
sure.instruments.keithley.Keithley2450 prop-	source_mode (pymeasure.instruments.yokogawa.Yokogawa7651
erty), 144	property), 216
source_current_delay_auto (pymea-	
sure.instruments.keithley.Keithley2450 prop-	property), 216
erty), 144	source_range (pymea-
source_current_range (pymea-	
sure.instruments.keithley.Keithley2400 prop-	erty), 151
erty), 138	source_range (pymea-
source_current_range (pymea-	sure.instruments.yokogawa.YokogawaGS200
sure.instruments.keithley.Keithley2450 prop-	property), 216
erty), 144	source_voltage (pymea-
source_current_range (pymea-	sure.instruments.keithley.Keithley2400 prop-
sure.instruments.yokogawa.Yokogawa7651	erty), 138
property), 216	source_voltage (pymea-
source_current_resistance_limit (pymea-	sure.instruments.keithley.Keithley2450 prop-
sure.instruments.keithley.Keithley6517B	erty), 144
property), 156	source_voltage (pymea-
source_delay (pymea-	sure.instruments.keithley.Keithley6517B
sure.instruments.keithley.Keithley2400 prop-	property), 156
erty), 138	source_voltage (pymea-
source_delay (pymea- sure instruments keithley Keithley 6221 prop-	sure.instruments.yokogawa.Yokogawa7651 property) 216
NUTE THIS IT WITHER IN KETTITLE VICENTIAL TO THE NEW TOTAL TO THE NEW TOTAL TO THE NEW TOTAL TOTAL TOTAL TOTAL TOTAL TOTAL TO THE NEW TOTAL TOTA	DIODELLYI, Z.IV)

- · · · · · · · · · · · · · · · · · · ·	ea- standard_devs (pymea-
sure.instruments.keithley.Keithley2450 pro erty), 144	op- sure.instruments.keithley.Keithley2450 prop- erty), 144
· · · · · · · · · · · · · · · · · · ·	ea- standard_event_enabled (pymea-
sure.instruments.keithley.Keithley2450 pro erty), 144	
source_voltage_range (pyma	
	op- sure.instruments.keithley.Keithley6221 prop- erty), 152
source_voltage_range (pyma	4156771777
sure.instruments.keithley.Keithley2450 pro	op- property), 81
erty), 144	START (pymeasure.instruments.agilent.agilentB1500.StaircaseSweepPostO
source_voltage_range (pyma	
sure.instruments.keithley.Keithley6517B	start() (pymeasure.experiment.experiment.Experiment
property), 156 source_voltage_range (pyma	<pre>method), 42 ea- start_buffer()</pre>
sure.instruments.yokogawa.Yokogawa7651	ea- start_buffer() (pymea- sure.instruments.keithley.Keithley2000
property), 216	method), 131
spacing (pymeasure.instruments.agilent.agilent4156.	
property), 80	sure.instruments.keithley.Keithley2400
span_frequency (pyma	
sure.instruments.advantest.advantestR3767C	
property), 66	sure.instruments.keithley.Keithley2450
special_channel (pyma	
sure.instruments.rohdeschwarz.sfm.SFM	start_buffer() (pymea-
property), 195	sure.instruments.keithley.Keithley2700
SPOT (pymeasure.instruments.agilent.agilentB1500.Me	
attribute), 102	start_buffer() (pymea-
square_dutycycle (pyma	·
sure.instruments.agilent.Agilent33220A	method), 152
property), 84	start_buffer() (pymea-
square_dutycycle (pyma	·
	op- method), 156
erty), 87	start_buffer() (pymea-
SR510 (class in pymeasure instruments.srs), 204	sure.instruments.signalrecovery.DSP7265 method), 203
SR570 (class in pymeasure.instruments.srs), 204 SR830 (class in pymeasure.instruments.srs), 205	
SR860 (class in pymeasure.instruments.srs), 203	start_frequency (pymea- sure.instruments.advantest.advantestR3767CG.AdvantestR3767C
$srq_{event\_enabled}$ (pymasure.thstruments.srs), 200	
- · · · · · · · · · · · · · · · · · · ·	op- start_frequency (pymea-
erty), 151	sure.instruments.agilent.Agilent8257D prop-
SRQ_mask (pymeasure.instruments.hp.HP3478A pro	v v i i
erty), 123	start_frequency (pymea-
STAIRCASE_SWEEP (pyma	
sure.instruments.agilent.agilentB1500.Meast	
attribute), 102	start_frequency (pymea-
staircase_sweep_source() (pyma	· · · · · · · · · · · · · · · · · · ·
sure.instruments.agilent.agilentB1500.SMU	property), 70
method), 98	start_power(pymeasure.instruments.agilent.Agilent8257D
StaircaseSweepPostOutput (class in pyma	
sure.instruments.agilent.agilentB1500), 103	
standard_devs (pyma	
sure.instruments.keithley.Keithley2400 pro	
erty), 139	start_sequence() (pymea-

sure.instruments.danfysik.Danfysik8500 method), 118	status_reg(pymeasure.instruments.rohdeschwarz.sfm.SFM property), 195
start_step_sweep() (pymea- sure.instruments.agilent.Agilent8257D	std_current (pymeasure.instruments.keithley.Keithley2400 property), 139
method), 68	std_current(pymeasure.instruments.keithley.Keithley2450
startup() (pymeasure.experiment.procedure.Procedure	property), 145
method), 44	std_resistance (pymea-
startup() (pymeasure.experiment.procedure.UnknownPr	**
method), 44	erty), 139
state (pymeasure.instruments.ami.AMI430 property),	•
106	sure.instruments.keithley.Keithley2450 prop-
status (pymeasure.instruments.agilent.agilentB1500.SMU	
property), 97	std_voltage(pymeasure.instruments.keithley.Keithley2400
status (pymeasure.instruments.bkprecision.BKPrecision9	
property), 115	std_voltage(pymeasure.instruments.keithley.Keithley2450
status (pymeasure.instruments.danfysik.Danfysik8500	property), 145
property), 118	step (pymeasure.instruments.agilent.agilent4156.VARX
status (pymeasure.instruments.hp.HP3478A property),	property), 81
126	step_points(pymeasure.instruments.agilent.Agilent8257D
status (pymeasure.instruments.Instrument property), 63	property), 68
status (pymeasure.instruments.keithley.Keithley2000	step_position (pymea-
property), 131	sure. instruments. an aheim automation. DPS eries Motor Controller
status (pymeasure.instruments.keithley.Keithley2260B	property), 108
property), 133	stepd (pymeasure.instruments.attocube.anc300.Axis
status (pymeasure.instruments.keithley.Keithley2450	property), 115
property), 145	steps_to_absolute() (pymea-
status (pymeasure.instruments.keithley.Keithley2600	sure. instruments. an aheim automation. DPS eries Motor Controller
property), 158	method), 109
status (pymeasure.instruments.keithley.Keithley2700	stepu (pymeasure.instruments.attocube.anc300.Axis
property), 148	property), 115
status (pymeasure.instruments.keithley.Keithley2750	stop (pymeasure.instruments.agilent.agilent4156.VARX
property), 157	property), 82
status (pymeasure.instruments.keithley.Keithley6221 property), 152	STOP (pymeasure.instruments.agilent.agilentB1500.StaircaseSweepPostOv attribute), 103
status (pymeasure.instruments.keithley.Keithley6517B	stop() (pymeasure.experiment.listeners.Recorder
property), 156	method), 43
status (pymeasure.instruments.keysight.KeysightDSOX11 property), 161	02Cop() (pymeasure.instruments.agilent.agilent4156.Agilent4156 method), 79
	stop() (pymeasure.instruments.anaheimautomation.DPSeriesMotorCont.
property), 163	method), 109
status (pymeasure.instruments.parker.ParkerGV6 property), 183	stop() (pymeasure.instruments.attocube.anc300.Axis method), 115
status (pymeasure.instruments.srs.SR510 property), 204	stop() (pymeasure.instruments.keysight.KeysightDSOX1102G method), 161
status() (pymeasure.instruments.keithley.Keithley2400 method), 139	stop() (pymeasure.instruments.ni.virtualbench.VirtualBench.FunctionGe method), 174
	Memoa), 174 Ostop() (pymeasure.instruments.ni.virtualbench.VirtualBench.MixedSigna
property), 118	method), 177
status_info_shown (pymea-	stop() (pymeasure.instruments.parker.ParkerGV6
sure.instruments.rohdeschwarz.sfm.SFM	method), 183
property), 195	stop_all() (pymeasure.instruments.attocube.anc300.ANC300Controller
status_preset() (pymea-	method), 114
sure.instruments.rohdeschwarz.sfm.SFM	stop_buffer() (pymea-
method) 195	sure instruments keithley Keithley2000

method), 131 stop_buffer() (pymeasure.instruments.keithley.Keithley2400	sweep_auto_abort() (pymea- sure.instruments.agilent.agilentB1500.AgilentB1500 method), 96
<pre>method), 139 stop_buffer()</pre>	sweep_status (pymea- sure.instruments.oxfordinstruments.ITC503
sure.instruments.keithley.Keithley2450 method), 145	property), 181 sweep_table (pymeasure.instruments.oxfordinstruments.ITC503
stop_buffer() (pymea- sure.instruments.keithley.Keithley2700	property), 181 sweep_time(pymeasure.instruments.agilent.Agilent8722ES
method), 148 stop_buffer() (pymea- sure.instruments.keithley.Keithley6221	property), 70 sweep_time (pymeasure.instruments.agilent.AgilentE4408B property), 70
<pre>method), 152 stop_buffer() (pymea-</pre>	sweep_time(pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200
sure.instruments.keithley.Keithley6517B method), 156	sweep_timing() (pymea- sure.instruments.agilent.agilentB1500.AgilentB1500
stop_frequency (pymea- sure.instruments.advantest.advantestR3767CG.A property), 66	method), 95 Ad <b>SwerepMAT6</b> 7CG (class in pymea- sure.instruments.agilent.agilentB1500), 102
stop_frequency (pymea- sure.instruments.agilent.Agilent8257D prop- erty), 68	
stop_frequency (pymea- sure.instruments.agilent.Agilent8722ES property), 70	synchronous_sweep_source() (pymea- sure.instruments.agilent.agilentB1500.SMU method), 99
stop_frequency (pymea- sure.instruments.agilent.AgilentE4408B property), 70	
stop_power (pymeasure.instruments.agilent.Agilent82571 property), 69	Osystem_setup (pymea- sure.instruments.keysight.KeysightDSOX1102G
stop_ramp() (pymeasure.instruments.danfysik.Danfysik8 method), 118	
stop_sequence() (pymea- sure.instruments.danfysik.Danfysik8500 method), 118	TabWidget (class in pymeasure.display.widgets), 56 target_current (pymeasure.instruments.ami.AMI430
<pre>stop_step_sweep()</pre>	property), 106 target_field (pymeasure.instruments.ami.AMI430
method), 69 StoppableQThread (class in pymeasure.display.thread), 54	property), 106  TDS2000 (class in pymeasure.instruments.tektronix), 213  TEDSetTemperature (pymea-
StringInput (class in pymeasure.display.inputs), 51 strip_chart_dat1 (pymeasure.instruments.srs.SR860	sure.instruments.thorlabs.ThorlabsPro8000 property), 214
<pre>property), 212 strip_chart_dat2 (pymeasure.instruments.srs.SR860 property), 212</pre>	TEDStatus (pymeasure.instruments.thorlabs.ThorlabsPro8000 property), 214 TelnetAdapter (class in pymeasure.adapters), 39
<pre>strip_chart_dat3 (pymeasure.instruments.srs.SR860</pre>	temperature (pymeasure.instruments.agilent.Agilent34450A property), 76
strip_chart_dat4 (pymeasure.instruments.srs.SR860 property), 213	temperature (pymeasure.instruments.fluke.Fluke7341 property), 120
subsystem_info (pymea- sure.instruments.rohdeschwarz.sfm.SFM property), 195	temperature(pymeasure.instruments.keithley.Keithley2000 property), 131 temperature_1 (pymea-
supply_current (pymeasure.instruments.ami.AMI430 property), 106	sure.instruments.oxfordinstruments.ITC503 property), 181

temperature_2 (pymea-	property), 95
sure. instruments. ox for dinstruments. ITC 503	$\verb timebase  (pymeasure. instruments. keysight. KeysightDSOX1102G $
property), 181	property), 161
temperature_3 (pymea-	timebase (pymeasure.instruments.srs.SR860 property),
sure. instruments. ox for dinstruments. ITC 503	213
property), 181	timebase_mode (pymea-
temperature_A (pymea-	sure.instruments.keysight.KeysightDSOX1102G
sure. instruments. lake shore. Lake Shore 331	property), 162
property), 166	timebase_offset (pymea-
temperature_B (pymea-	sure.instruments.keysight.KeysightDSOX1102G
sure. instruments. lake shore. Lake Shore 331	property), 162
property), 166	timebase_range (pymea-
temperature_digits (pymea-	sure.instruments.keysight.KeysightDSOX1102G
sure.instruments.keithley.Keithley2000 prop-	property), 162
erty), 131	timebase_scale (pymea-
temperature_error (pymea-	sure.instruments.keysight.KeysightDSOX1102G
sure.instruments.ox for dinstruments.ITC 503	property), 162
property), 182	timebase_setup() (pymea-
temperature_nplc (pymea-	sure.instruments.keysight.KeysightDSOX1102G
sure.instruments.keithley.Keithley2000 prop-	method), 162
erty), 131	to_dict() (pymeasure.instruments.agilent.agilentB1500.QueryLearn
temperature_reference (pymea-	static method), 100
sure.instruments.keithley.Keithley2000 prop-	trace() (pymeasure.instruments.agilent.AgilentE4408B
erty), 131	method), 70
temperature_setpoint (pymea-	${\tt trace\_1} \ (pymeasure. instruments. advantest. advantestR3767CG. AdvantestR376CG. AdvantestR376$
sure.instruments.oxfordinstruments.ITC503	property), 66
property), 182	trace_df() (pymeasure.instruments.agilent.AgilentE4408B
text_enabled (pymea-	method), 70
sura instruments kaithley Kaithley 2700 nron	
sure.instruments.keithley.Keithley2700 prop-	trace_marker (pymea-
erty), 148	sure.instruments.anritsu.AnritsuMS9710C
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208	sure.instruments.anritsu.AnritsuMS9710C property), 111
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymea-
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymea-	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C  property), 112
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs)	sure.instruments.anritsu.AnritsuMS9710C property), 111 trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C property), 112 trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214	sure.instruments.anritsu.AnritsuMS9710C property), 111 trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C property), 112 trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCM	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C  property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL  property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSup
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C  property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL  property), 200  lotteacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSup)  property), 178
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102 time (pymeasure.instruments.Mock property), 63	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  fotheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFM	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  fotleacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSuppoperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  lotleacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property)	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C  property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL  property), 200  Notheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400  method), 139  triad() (pymeasure.instruments.keithley.Keithley2450  method), 145  triad() (pymeasure.instruments.keithley.Keithley2700
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105	sure.instruments.anritsu.AnritsuMS9710C  property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C  property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL  property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSuppoperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400  method), 139  triad() (pymeasure.instruments.keithley.Keithley2450  method), 145  triad() (pymeasure.instruments.keithley.Keithley2700  method), 148
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  fotheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.signalrecovery.DSP7265	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.signalrecovery.DSP7265 property), 203	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152  trigger (pymeasure.instruments.hp.HP3478A prop-
erty), 148 theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCMattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.signalrecovery.DSP7265 property), 203 time_constant (pymeasure.instruments.srs.SR510	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152  trigger (pymeasure.instruments.hp.HP3478A property), 126
erty), 148  theta (pymeasure.instruments.srs.SR830 property), 208  theta (pymeasure.instruments.srs.SR860 property), 213  ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214  ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214  TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102  time (pymeasure.instruments.Mock property), 63  time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195  time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105  time_constant (pymeasure.instruments.signalrecovery.DSP7265property), 203  time_constant (pymeasure.instruments.srs.SR510property), 204	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152  trigger (pymeasure.instruments.hp.HP3478A prop-
theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.signalrecovery.DSP7265property), 203 time_constant (pymeasure.instruments.srs.SR510property), 204 time_constant (pymeasure.instruments.srs.SR830	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152  trigger (pymeasure.instruments.hp.HP3478A property), 126  trigger() (pymeasure.instruments.agilent.Agilent33220A method), 84
theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.signalrecovery.DSP7265 property), 203 time_constant (pymeasure.instruments.srs.SR510 property), 204 time_constant (pymeasure.instruments.srs.SR830 property), 208	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymea- sure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Motheacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSup, property), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152  trigger (pymeasure.instruments.hp.HP3478A property), 126  trigger() (pymeasure.instruments.agilent.Agilent33220A
theta (pymeasure.instruments.srs.SR830 property), 208 theta (pymeasure.instruments.srs.SR860 property), 213 ThorlabsPM100USB (class in pymeasure.instruments.thorlabs), 214 ThorlabsPro8000 (class in pymeasure.instruments.thorlabs), 214 TIME (pymeasure.instruments.agilent.agilentB1500.ADCNattribute), 102 time (pymeasure.instruments.Mock property), 63 time (pymeasure.instruments.rohdeschwarz.sfm.SFMproperty), 195 time_constant (pymeasure.instruments.ametek.Ametek7270 property), 105 time_constant (pymeasure.instruments.signalrecovery.DSP7265property), 203 time_constant (pymeasure.instruments.srs.SR510property), 204 time_constant (pymeasure.instruments.srs.SR830property), 208	sure.instruments.anritsu.AnritsuMS9710C property), 111  trace_marker_center (pymeasure.instruments.anritsu.AnritsuMS9710C property), 112  trace_mode (pymeasure.instruments.rohdeschwarz.fsl.FSL property), 200  Mothacking (pymeasure.instruments.ni.virtualbench.VirtualBench.PowerSupproperty), 178  triad() (pymeasure.instruments.keithley.Keithley2400 method), 139  triad() (pymeasure.instruments.keithley.Keithley2450 method), 145  triad() (pymeasure.instruments.keithley.Keithley2700 method), 148  triad() (pymeasure.instruments.keithley.Keithley6221 method), 152  trigger (pymeasure.instruments.hp.HP3478A property), 126  trigger() (pymeasure.instruments.agilent.Agilent33220A method), 84  trigger() (pymeasure.instruments.agilent.Agilent33500

	() (pymeasure.instruments.keithley.Keithley2450 method), 145	trigger	_state sure.instruments.agilent.Agilent33220	(pymea- )A
	() (pymeasure.instruments.keithley.Keithley6221 method), 152		property), 84 e_lines()	(рутеа-
trigger(	() (pymeasure.instruments.keithley.Keithley65171 method), 156		sure.instruments.ni.virtualbench.Virtu method), 170	4.5
trigger_		TV_coun	try (pymeasure.instruments.rohdeschv	varz.sfm.SFM
	sure.instruments.keithley.Keithley2000 prop-		property), 186	
	erty), 131	TV_stan	${\tt dard} (pymeasure. instruments. rohdesch$	hwarz.sfm.SFM
trigger_			property), 187	
	sure.instruments.keithley.Keithley2400 prop-	1.1		
	erty), 139	U		
trigger_			_front() (pymeasure.instruments.s	srs.SR570
	sure.instruments.keithley.Keithley2000 prop-		method), 205	
trigger_	erty), 131 _delay (pymea-	unique_	filename() (in module	рутеа-
	sure.instruments.keithley.Keithley2400 prop-		sure.experiment.results), 48	
	erty), 139	unit (py	measure.instruments.fluke.Fluke7341 ¡ 120	property),
	_immediately() (pymea-	unit (	pymeasure.instruments.lakeshore.Lake	Shore425
	sure.instruments.keithley.Keithley2400	unit (	property), 167	Shore+25
	method), 139	units(n	ymeasure.instruments.fwbell.FWBell50	080 prop-
trigger_	_immediately() (pymea-		erty), 121	r r r
	sure.instruments.keithley.Keithley6221	units	(pymeasure.instruments.newport.es	p300.Axis
	method), 152		property), 168	
	_immediately() (pymea-		Procedure (class in	рутеа-
	sure.instruments.keithley.Keithley6517B		sure.experiment.procedure), 44	
	method), 156	update(	2.7	Crosshairs
	_on_bus() (pymea-		method), 50	
	sure.instruments.keithley.Keithley2400 method), 139	update_		(pymea-
	_on_bus() (pymea-		sure.display.curves.ResultsCurve	method),
	sure.instruments.keithley.Keithley6221		50	(
	method), 152	upaa ce_	estimates()	(pymea-
	_on_bus() (pymea-		sure.display.widgets.EstimatorWidget 55	meinoa),
	sure.instruments.keithley.Keithley6517B	update_		(рутеа-
	method), 156	up uu co_	sure.experiment.experiment.Experime	* *
trigger_	_on_external() (pymea-		method), 42	
	sure.instruments.keithley.Keithley2400	update_	parameter() (pymeasure.display.inp	puts.Input
	method), 139		method), 51	
	_on_external() (pymea-	update_	pcolor()	(рутеа-
	sure.instruments.keithley.Keithley6221		sure.experiment.experiment.Experiment.expe	nt
	method), 152		method), 42	
	_ramp_to_level() (pymea-	update_	-	(рутеа-
	sure.instruments.yokogawa.YokogawaGS200 method), 217		sure.experiment.experiment.Experime	nt
trigger_			method), 42	(
	sure.instruments.agilent.Agilent33220A	upaate_	status() sure.experiment.workers.Worker	(pymea-
	property), 84		47	method),
trigger_	* * · · · · · · · · · · · · · · · · · ·	use abs	olute_position()	(рутеа-
	sure.instruments.agilent.Agilent33500 prop-	~5C_ab3	sure.instruments.parker.ParkerGV6	method),
	erty), 87		183	,
trigger_	_ ·	use_ext	ernal_source	(рутеа-
	sure.instruments.agilent.AgilentE4980 prop-		sure.instruments.rohdeschwarz.sfm.Sc	
	erty), 72		property), 197	

<pre>use_front_terminals()</pre>	ymea-	values() (pymeasure.instruments.Instrument method), 63
method), 139		values() (pymeasure.instruments.keysight.KeysightDSOX1102G
	утеа-	method), 162
sure.instruments.keithley.Keithley2450 method), 145		values() (pymeasure.instruments.lakeshore.LakeShoreUSBAdapter method), 164
	утеа-	values() (pymeasure.instruments.rohdeschwarz.sfm.Sound_Channel
sure.instruments.keithley.Keithley2400		method), 197
method), 139		VAR1 (class in pymea-
<pre>use_rear_terminals()</pre>	ymea-	sure.instruments.agilent.agilent4156), 80
sure.instruments.keithley.Keithley2450		VAR2 (class in pymea-
method), 145		sure.instruments.agilent.agilent4156), 81
		VARD (class in pymea-
sure.instruments.parker.ParkerGV6 met	thod),	sure.instruments.agilent.agilent4156), 81
184		VARX (class in pymea-
V		sure.instruments.agilent.agilent4156), 81
V		VectorParameter (class in pymea-
<pre>validate_auto_range_terminal()</pre>	ymea-	sure.experiment.parameters), 46
sure.instruments.ni.virtual bench. Virtual Bench.	ench.Dig	i giansian hemeasure.instruments.attocube.anc300.ANC300Controller
method), 172		property), 114
		video_bandwidth (pymea-
	ench.Mi.	ixedSignal@wcellostropnents.rohdeschwarz.fsl.FSL prop-
method), 177		erty), 200
		VirtualBench (class in pymea-
		owerSupplysure.instruments.ni.virtualbench), 169
method), 178		VirtualBench.DigitalInputOutput(class in pymea-
	ymea-	sure.instruments.ni.virtualbench), 169
	ench.Dig	WindMalBench.DigitalMultimeter (class in pymea-
method), 172		sure.instruments.ni.virtualbench), 171
**		VirtualBench.FunctionGenerator (class in pymea-
		igitalInputOutpunstruments.ni.virtualbench), 172
method), 170		VirtualBench.MixedSignalOscilloscope (class in
	ymea-	pymeasure.instruments.ni.virtualbench), 174
sure.instruments.ni.virtualbench.VirtualB	encn.Dış	igliatMutliBench.PowerSupply (class in pymea- sure.instruments.ni.virtualbench), 178
static method), 172		
		VirtualBench_Direct (class in pymea- ixedSignal <b>@wwelloswope</b> ents.ni.virtualbench), 180
static method), 178	encn.mi.	VISAAdapter (class in pymeasure.adapters), 36
values() (pymeasure.adapters.Adapter method), 3		vision_average_enabled (pymea-
values() (pymeasure.adapters.FakeAdapter method), svalues() (pymeasure.adapters.FakeAdapter method)		sure.instruments.rohdeschwarz.sfm.SFM
values() (pymeasure.adapters.PrologixAa		property), 195
method), 36		vision_balance (pymea-
values() (pymeasure.adapters.SerialAdapter mei		sure.instruments.rohdeschwarz.sfm.SFM
34	mou),	property), 195
values() (pymeasure.adapters.TelnetAdapter met	thod).	vision_carrier_enabled (pymea-
40	,,	sure.instruments.rohdeschwarz.sfm.SFM
values() (pymeasure.adapters.VISAAdapter med	thod).	property), 195
37		vision_carrier_frequency (pymea-
values() (pymeasure.adapters.VXII1Adapter met		sure.instruments.rohdeschwarz.sfm.SFM
39	,,	property), 195
values() (pymeasure.instruments.anaheimautoma	tion.DP	
method), 109		sure.instruments.rohdeschwarz.sfm.SFM
values() (pymeasure.instruments.fwbell.FWBel	115080	property), 195
method), 121		vision_clamping_enabled (pymea-

	re.instruments.rohdeschwarz.sfm.SFM operty), 195	voltage	_ac (pymeasure.instruments.hp.HP property), 123	34401A
sui	amping_mode (pyr. re.instruments.rohdeschwarz.sfm.SFM operty), 195	<i>nea</i> - voltage	_ac_auto_range ( sure.instruments.agilent.Agilent34450A property), 76	рутеа-
sur pro	ecorrection_enabled (pyname.instruments.rohdeschwarz.sfm.SFM operty), 196		sure.instruments.keithley.Keithley2000 erty), 131	pymea- prop-
sui	sidual_carrier_level (pyn re.instruments.rohdeschwarz.sfm.SFM operty), 196	<i>nea-</i> voltage	_ac_digits ( sure.instruments.keithley.Keithley2000 erty), 131	pymea- prop-
vision_sid	deband_filter_enabled (pyname.instruments.rohdeschwarz.sfm.SFM operty), 196	<i>nea-</i> voltage	_ac_nplc (sure.instruments.keithley.Keithley2000 erty), 132	pymea- prop-
sui	deosignal_enabled (pyn re.instruments.rohdeschwarz.sfm.SFM operty), 196	<i>nea-</i> voltage	_ac_range ( sure.instruments.agilent.Agilent34450A property), 76	рутеа-
VMU sui		<i>nea-</i> voltage 250A		pymea- prop-
pro	operty), 76 vmeasure.instruments.agilent.agilentB150	voltage	_ac_reference (	pymea- prop-
voltage	tribute), 102 (pymeasure.instruments.ametek.Ametek? operty), 105	<i>7270</i> voltage	erty), 132 _ac_resolution sure.instruments.agilent.Agilent34450A	рутеа-
voltage (pro	pymeasure.instruments.attocube.anc300. operty), 115	voltage	property), 76 _auto_range	рутеа-
pro	ymeasure.instruments.bkprecision.BKPre operty), 116 ymeasure.instruments.deltaelektronika.SN		property), 76	
provoltage (p	operty), 120 pymeasure.instruments.keithley.Keithley2		property), 72 _dc (pymeasure.instruments.hp.HP	
voltage (py	operty), 131 ymeasure.instruments.keithley.Keithley22 operty), 133	<i>60B</i> voltage	property), 123 _digits ( sure.instruments.keithley.Keithley2000	pymea- prop-
voltage (pro	pymeasure.instruments.keithley.Keithley2 operty), 139	voltage	<pre>erty), 132 _filter_count</pre>	рутеа-
pro	pymeasure.instruments.keithley.Keithley2 operty), 145 ymeasure.instruments.keithley.Keithley65		sure.instruments.keithley.Keithley2450 erty), 145 filter_type	prop- pymea-
pro voltage(py	operty), 156 ymeasure.instruments.keysight.KeysightN	5767A	sure.instruments.keithley.Keithley2450 erty), 145	prop-
voltage (py	operty), 163 ymeasure.instruments.Mock property), 63 ymeasure.instruments.signalrecovery.DSI		_high ( sure.instruments.agilent.Agilent33220A property), 84	рутеа-
provoltage_1	operty), 203 (pymeasure.instruments.razorbill.razorbi operty), 186	voltage		pymea- prop-
voltage_2(	(pymeasure.instruments.razorbill.razorbi operty), 186		_limit (pymeasure.instruments.ami.a property), 106	
pro	c (pymeasure.instruments.agilent.Agilent. operty), 72 c (pymeasure instruments gailent Agilent		sure.instruments.yokogawa.YokogawaG	(pymea- S200
	C (pymeasure.instruments.agilent.Agilent. operty), 76		property), 217 _low(pymeasure.instruments.agilent.Ag	ilent33220A

	property), 84	VSU	(class	,	in	рутеа-
voltage_	$\_1$ ow (pymeasure.instruments.agilent.Agilent335	00	sure.instrum	ents.agileni	t.agilent4156).	, 82
	property), 88	VXI11	Adapter (class	in pymeasu	ire.adapters), (	38
voltage_	_name					
	sure.instruments.agilent.agilent4156.SMU	W				
	property), 80	wait(	) (pymeasure.in	struments.a	ınritsu.Anritsu	MS9710C
voltage_	_name (pymea-		method), 112			
	sure.instruments.agilent.agilent4156.VMU	wait	for_buffer()			(рутеа-
	property), 82				y.Keithley2000	
voltage_	_name		method), 132			
	sure.instruments.agilent.agilent4156.VSU	wait_	for_buffer()			(рутеа-
	property), 82				y.Keithley2400	- ·
voltage <u>.</u>	_nplc (pymea-		method), 140			
	sure.instruments.keithley.Keithley2000 prop-	wait_	for_buffer()			(рутеа-
	erty), 132		sure.instrum	ents.keithle	y.Keithley2450	0
voltage_	_nplc (pymea-		method), 145	5		
	sure.instruments.keithley.Keithley2400 prop-	wait_	for_buffer()			(рутеа-
	erty), 140		sure.instrum	ents.keithle	y.Keithley2700	9
voltage_			method), 148	8		
	sure.instruments.keithley.Keithley2450 prop-	wait_	for_buffer()			(рутеа-
	erty), 145				y.Keithley622	1
voltage <sub>-</sub>	_ · · · · · · · · · · · · · · · · · · ·		method), 152			
	sure.instruments.keithley.Keithley6517B	wait_	for_buffer()			(рутеа-
	property), 156				y.Keithley6517	7B
	_output_off_state (pymea-	_	method), 150			
	sure.instruments.keithley.Keithley2450 prop-	wait_	for_buffer()		n an	(pymea-
	erty), 145				recovery.DSP7	/265
voltage <sub>-</sub>	_range (pymea- sure.instruments.agilent.Agilent34450A		method), 203			gp020
	property), 76	wait_	for_buffer()		e.instruments.s	srs.SR830
voltage_			method), 208			(
	sure.instruments.keithley.Keithley2000 prop-		for_completi.			(pymea-
	erty), 132				ташотаноп.	DPSeriesMotorController
voltage_		wai+	method), 109 for_current(			(рутеа-
	sure.instruments.keithley.Keithley2400 prop-				ik.Danfysik850	- ·
	erty), 140		method), 118		in.Dungysinos	50
voltage_	_range	wait	for_data()			(рутеа-
	sure.instruments.keithley.Keithley2450 prop-	_		ient.experin	nent.Experime	- ·
	erty), 145		method), 42	1		
voltage <u>.</u>	•	wait_	for_holding(	)		(рутеа-
	sure.instruments.keithley.Keithley6517B		sure.instrum	ents.ami.Al	MI430 method	), 106
	property), 156	wait_	for_ready()			(рутеа-
voltage_	-		sure.instrum	ents.danfys	ik.Danfysik850	00
	sure.instruments.keysight.KeysightN5767A		method), 118	8		
	property), 163	wait_	for_srq() (py	measure.aa	lapters.Prolog	ixAdapter
_	_reference (pymea-		method), 36			
	sure.instruments.keithley.Keithley2000 prop-	wait_	_	(pymeasure	e.adapters.VIS	'AAdapter
	erty), 132		method), 37			
_	_resolution (pymea-	wait_	for_stop()		222	(pymea-
	sure.instruments.agilent.Agilent34450A			ents.newpo	rt.esp300.Axis	method),
	<pre>property), 76 _setpoint (pymea-</pre>		168			(
	sure.instruments.keithley.Keithley2260B	wait_	for_sweep()		. Amita MCC	(pymea-
	property), 133				ı.AnritsuMS97	10C
	I . I / / /		method), 112	∠		

<pre>wait_for_temperature()</pre>	sure.instruments.keysight.KeysightDSOX1102G
sure. instruments. lake shore. Lake Shore 331	property), 162
method), 166	waveform_points_mode (pymea-
<pre>wait_for_temperature()</pre>	sure.instruments.keysight.KeysightDSOX1102G
sure. instruments. ox for dinstruments. ITC 503	property), 162
method), 182	waveform_preamble (pymea-
<pre>wait_for_trigger()</pre> <pre>(pymea-</pre>	sure.instruments.keysight.KeysightDSOX1102G
sure.instruments.agilent.Agilent33220A	property), 162
method), 84	waveform_ranging (pymea-
<pre>wait_for_trigger()</pre>	sure.instruments.keithley.Keithley6221 prop-
sure.instruments.agilent.Agilent33500 method),	erty), 153
88	waveform_source (pymea-
<pre>wait_time() (pymeasure.instruments.agilent.agilentB15</pre>	00.AgilentB <b>&amp;&amp;O</b> linstruments.keysight.KeysightDSOX1102G property), 162
	waveform_start() (pymea-
sure.instruments.agilent.agilentB1500), 103	sure.instruments.keithley.Keithley6221
wave (pymeasure.instruments.Mock property), 63	method), 153
	waveform_use_phasemarker (pymea-
sure.instruments.keithley.Keithley6221	sure.instruments.keithley.Keithley6221 prop-
method), 152	erty), 153
waveform_amplitude (pymea-	* * * *
sure.instruments.keithley.Keithley6221 prop-	property), 214
erty), 152	wavelength_center (pymea-
waveform_arm() (pymea-	-
sure.instruments.keithley.Keithley6221	property), 112
method), 153	wavelength_marker_value (pymea-
waveform_data (pymea-	sure.instruments.anritsu.AnritsuMS9710C
sure.instruments.keysight.KeysightDSOX1102G	
property), 162	wavelength_max (pymea-
waveform_duration_cycles (pymea-	sure.instruments.thorlabs.ThorlabsPM100USB
sure.instruments.keithley.Keithley6221 prop-	property), 214
erty), 153	wavelength_min (pymea-
<pre>waveform_duration_set_infinity() (pymea-</pre>	-
sure.instruments.keithley.Keithley6221	property), 214
method), 153	wavelength_span (pymea-
waveform_duration_time (pymea-	
sure.instruments.keithley.Keithley6221 prop-	
erty), 153	wavelength_start (pymea-
waveform_dutycycle (pymea-	sure.instruments.anritsu.AnritsuMS9710C
sure.instruments.keithley.Keithley6221 prop-	property), 112
erty), 153	wavelength_stop (pymea-
waveform_format (pymea-	sure.instruments.anritsu.AnritsuMS9710C
sure.instruments.keysight.KeysightDSOX1102G	
property), 162	wavelength_value_in (pymea-
waveform_frequency (pymea-	sure.instruments.anritsu.AnritsuMS9710C
sure.instruments.keithley.Keithley6221 prop-	property), 112
erty), 153	wavelengths (pymeasure.instruments.anritsu.AnritsuMS9710C
waveform_function (pymea-	property), 112
sure.instruments.keithley.Keithley6221 prop-	
erty), 153	property), 140
waveform_offset (pymea-	wires (pymeasure.instruments.keithley.Keithley2450
sure.instruments.keithley.Keithley6221 prop-	property), 146
erty), 153	Worker (class in pymeasure.experiment.workers), 46
waveform_points (pymea-	
pymea	= () (p j pro 1 su 1 su pro 1 montou ), 52

write() (pymeasure.adapters.FakeAdapter method), 33 write() (pymeasure.adapters.PrologixAdapter method),	xy (pymeasure.instruments.ametek.Ametek7270 prop-
36	erty), 105 xy (pymeasure.instruments.signalrecovery.DSP7265
write() (pymeasure.adapters.SerialAdapter method), 34	property), 203
write() (pymeasure.adapters.TelnetAdapter method), 40	xy (pymeasure.instruments.srs.SR830 property), 208
write() (pymeasure.adapters.VISAAdapter method), 38	, <sub>(F</sub> )
write() (pymeasure.adapters.VXII1Adapter method),	Y
39 write() (pymeasure.instruments.agilent.agilentB1500.SM	y (pymeasure.instruments.ametek.Ametek7270 property),
method), 97	y (pymeasure.instruments.signalrecovery.DSP7265 prop-
write() (pymeasure.instruments.anaheimautomation.DPS	GeriesMotorGoutroffer
method), 109	y (pymeasure.instruments.srs.SR830 property), 208
write() (pymeasure.instruments.attocube.adapters.Attocumethod), 113	by Consoleration by Con
write() (pymeasure.instruments.danfysik.DanfysikAdapte	r erty), 105
method), 116	y2 (pymeasure.instruments.ametek.Ametek7270 prop-
write() (pymeasure.instruments.Instrument method), 63	erty), 105
write() (pymeasure.instruments.keysight.KeysightDSOXI method), 163	•
write() (pymeasure.instruments.lakeshore.LakeShoreUSE	sure.instruments.yokogawa), 215 AddidgawaGS200 (class in pymea-
method), 165	sure.instruments.yokogawa), 216
write() (pymeasure.instruments.ni.virtualbench.VirtualBenchod), 170	property), 182
write() (pymeasure.instruments.parker.ParkerGV6 method), 184	Z
write_binary_values() (pymea-	
sure.adapters.PrologixAdapter method), 36	zero() (pymeasure.instruments.ami.AMI430 method), 107
write_binary_values() (pymea-	zero() (pymeasure.instruments.newport.esp300.Axis method), 168
sure.adapters.SerialAdapter method), 34 write_binary_values() (pymea-	zero_probe() (pymea-
sure.adapters.VISAAdapter method), 38	sure.instruments.lakeshore.LakeShore425
write_binary_values() (pymea-	method), 167
sure.instruments.lakeshore.LakeShoreUSBAdapte method), 165	er
write_raw() (pymeasure.adapters.VXI11Adapter	
method), 39	
<pre>writeAO() (in module pymeasure.instruments.comedi),</pre>	
65	
Χ	
x (pymeasure.instruments.ametek.Ametek7270 property), 105	
x (pymeasure.instruments.signalrecovery.DSP7265 prop-	
erty), 203	
x (pymeasure.instruments.srs.SR830 property), 208	
x (pymeasure.instruments.srs.SR860 property), 213	
x1 (pymeasure.instruments.ametek.Ametek7270 property), 105	
x2 (pymeasure.instruments.ametek.Ametek7270 property), 105	
xpointer (pymeasure.instruments.oxfordinstruments.ITC5 property), 182	03