

# API Documentation

API Documentation

February 19, 2014

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Module GPIBWrap</b>	<b>2</b>
1.1 Functions . . . . .	2
1.2 Variables . . . . .	2
1.3 Class GPIBInst . . . . .	2
1.3.1 Methods . . . . .	3
1.3.2 Properties . . . . .	3
1.4 Class HP_3457A . . . . .	4
1.4.1 Methods . . . . .	4
1.4.2 Properties . . . . .	5
1.5 Class HP_3325A . . . . .	5
1.5.1 Methods . . . . .	5
1.5.2 Properties . . . . .	7
1.6 Class HP_8903E . . . . .	7
1.6.1 Methods . . . . .	7
1.6.2 Properties . . . . .	8
1.7 Class HP_3582A . . . . .	8
1.7.1 Methods . . . . .	9
1.7.2 Properties . . . . .	11
1.8 Class TEK_710A . . . . .	11
1.8.1 Methods . . . . .	11
1.8.2 Properties . . . . .	12

# 1 Module GPIBWrap

(section) GPIBWrap.py - A Python Interface to the GPIBlib Instrument Library.

The combination of Python, the instrument-object based C++ GPIBlib instrument control library, NumPy and MatPlotLib provides a quite useful way of building measurement applications for computer controlled instruments.

Currently implemented instruments are:

HP\_3325A Synthesiser/Function Generator HP\_3457A Multimeter HP\_3582A FFT Spectrum Analyser  
HP\_8903E Distortion Analyser TEK\_710A Transient Digitizer (partially)

Author: Nick Glazzard 2012

## 1.1 Functions

`argerr()`

`callerr()`

`eu(v, u)`

`frequits(fu)`

`ampunits(au)`

`chancodes(ch)`

`couplecodes(cu)`

## 1.2 Variables

Name	Description
lib	<b>Value:</b> <code>cdll.LoadLibrary('./GPIBlib.so')</code>

## 1.3 Class GPIBInst



Base class for GPIB instruments with functions common to all instruments.

## 1.3.1 Methods

<code>__init__(self)</code>
-----------------------------

x. <code>__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature
-------------------------------------------------------------------------------------------

Overrides: object. <code>__init__</code> <code>__exit__</code> (inherited documentation)
------------------------------------------------------------------------------------------

<code>set_obj(self, obj)</code>
---------------------------------

<code>set_print_error(self, b)</code>
---------------------------------------

Allow (b=1) or suppress (b=0) error messages.
-----------------------------------------------

<code>set_print_gpib(self, b)</code>
--------------------------------------

Log (b=1) or not (b=0) all raw GPIB traffic.
----------------------------------------------

<code>set_display_status(self, b)</code>
------------------------------------------

Show status (b=1) or not (b=0) after each GPIB command.
---------------------------------------------------------

<code>set_dump_gpib(self, b)</code>
-------------------------------------

Dump a representation (b=1) or not (b=0) of binary GPIB read data.
--------------------------------------------------------------------

<code>reset_bus(self)</code>
------------------------------

Force a reset on the GPIB bus.
--------------------------------

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

## 1.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 1.4 Class HP\_3457A



HP 3457A Multimeter Object.

### 1.4.1 Methods

**\_\_init\_\_**(*self*, *a*, *g*)

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

**check\_open**(*self*)

Check the device is open. Return True if it is.

**set\_fast\_mode**(*self*, *b*)

Start using the HP 3457A for 3.5 digit measurements as fast as possible. If b=1, apply fast mode to AC measurements also (Only recommended for signals >400Hz).

**clear\_fast\_mode**(*self*)

End fast mode. Revert to high precision mode.

**set\_function**(*self*, *f*, *v*)

Set Function to f and Range to v. f may be: 'dcv', 'acv', 'acdcv', 'ohm', 'ohm4', 'dci', 'aci', 'acdc', 'freq', 'period' v depends on f. For volts functions: 'auto', '30mv', '300mv', '3v', '30v', '300v' For ohm functions: 'auto', '30r', '300r', '3k', '30k', '300k', '3m', '30m', '3g' For time functions: 'auto'

**read**(*self*)

Read the last measurement from the HP 3457A.

*Inherited from GPIBWrap.GPIBInst(Section 1.3)*

reset\_bus(), set\_display\_status(), set\_dump\_gpib(), set\_obj(), set\_print\_error(), set\_print\_gpib()

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

**1.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**1.5 Class HP\_3325A**

HP 3325A Synthesiser/Function Generator Object.

**1.5.1 Methods**

<code>__init__(self, a, g)</code>
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature
Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)

<code>check_open(self)</code>
Check the device is open. Return True if it is.

<code>check_error(self)</code>
Check the last HP_3325A specific command executed OK. Return True if so.

<code>set_function(self, sw)</code>
Set the output waveform to sw. sw may be: 'dc', 'sine', 'square', 'triangle', 'ramp_up', 'ramp_down'

**set\_frequency**(*self*, *freq*, *sunits*)

Set the output frequency to *freq* (in *sunits*). *sunits* may be: 'Hz', 'kHz', 'MHz'

**set\_amplitude**(*self*, *amp*, *sunits*)

Set the output amplitude to *amp* (in *sunits*). *sunits* may be: 'v', 'mv', 'v\_pp', 'mv\_pp', 'v\_rms', 'mv\_rms', 'dbm'

**set\_dc\_offset**(*self*, *dc*, *sunits*)

Set the output DC offset to *dc* (in *sunits*). *sunits* may be: 'v', 'mv'

**set\_phase**(*self*, *degrees*)

Set the phase offset to *degrees* degrees.

**set\_sweep\_start\_frequency**(*self*, *freq*, *sunits*)

Set the frequency at the start of a sweep (in *sunits*). *sunits* may be: 'Hz', 'kHz', 'MHz'

**set\_sweep\_end\_frequency**(*self*, *freq*, *sunits*)

Set the frequency at the end of a sweep (in *sunits*). *sunits* may be: 'Hz', 'kHz', 'MHz'

**set\_sweep\_marker\_frequency**(*self*, *freq*, *sunits*)

Set the frequency at which to output the marker signal during a sweep (in *sunits*). *sunits* may be: 'Hz', 'kHz', 'MHz'

**set\_sweep\_mode**(*self*, *ssweep\_mode*)

Set the sweep mode to 'lin' or 'log'.

**set\_sweep\_time**(*self*, *secs*)

Set the time over which to do a sweep (in seconds).

**start\_sweep\_single**(*self*, *waitfin*)

Do a single sweep. If *waitfin*=1, wait for it to end.

**start\_sweep\_continuous**(*self*)

Start sweeping continuously.

<b>use_amplitude_modulation</b> ( <i>self</i> , <i>yes</i> )
--------------------------------------------------------------

Turn amplitude modulation on (1) or off (0).
----------------------------------------------

<b>use_phase_modulation</b> ( <i>self</i> , <i>yes</i> )
----------------------------------------------------------

Turn phase modulation on (1) or off (0).
------------------------------------------

**Inherited from GPIBWrap.GPIBInst(Section 1.3)**

reset\_bus(), set\_display\_status(), set\_dump\_gpib(), set\_obj(), set\_print\_error(),  
set\_print\_gpib()

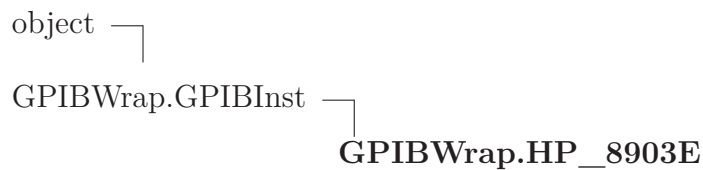
**Inherited from object**

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
\_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
\_\_str\_\_(), \_\_subclasshook\_\_()

### 1.5.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 1.6 Class HP\_8903E



HP 8903E Distortion Analyzer Object.

### 1.6.1 Methods

<b>__init__</b> ( <i>self</i> , <i>a</i> , <i>g</i> )
-------------------------------------------------------

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

Overrides: object.__init__ extit(inherited documentation)
-----------------------------------------------------------

<b>check_open</b> ( <i>self</i> )
-----------------------------------

Check the device is open. Return True if it is.
-------------------------------------------------

<b>get_ac_volts</b> ( <i>self</i> )
-------------------------------------

Measure the AC voltage present at the input.
----------------------------------------------

<b>get_distortion</b> ( <i>self</i> )
---------------------------------------

Measure the distortion of the signal present at the input.
------------------------------------------------------------

### *Inherited from GPIBWrap.GPIBInst(Section 1.3)*

reset\_bus(), set\_display\_status(), set\_dump\_gpib(), set\_obj(), set\_print\_error(),  
set\_print\_gpib()

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
\_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
\_\_str\_\_(), \_\_subclasshook\_\_()

#### 1.6.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 1.7 Class HP\_3582A



HP 3582A Spectrum Analyzer Object.



## 1.7.1 Methods

**\_\_init\_\_**(*self*, *a*, *g*)

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

**check\_open**(*self*)

Check the device is open. Return True if it is.

**set\_input\_mode**(*self*, *c*)

Set which channels to use. c may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2', 'both', 'chboth'

**coupling**(*self*, *c*, *m*)

Set the coupling mode for channel c to mode m. c may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2', 'both', 'chboth' m may be: 'ac':1, 'dc'

**set\_sensitivity**(*self*, *c*, *v*)

Set the sensitivity of channel c to v volts rms. c may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2', 'both', 'chboth' v is a voltage in the range 0.0 (30mV lowest range) to 30.0.

**set\_sensitivity\_db**(*self*, *c*, *v*)

Set the sensitivity of channel c to v volts rms in dB wrt 1V. c may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2', 'both', 'chboth' v is dBV in the range -50.0 to 30.0.

**set\_frequency**(*self*, *m*, *span*, *freq*)

Set the frequency axis of the spectrum. The use of span and freq depends on mode m. 'full': 0->25kHz [span] and [freq] ignored. 'zero\_start': 0->[span], [freq] ignored. 'set\_start': [freq]->[freq]+[span] 'set\_center': [freq]-[span]/2 -> [freq]+[span]/2

**display**(*self*, *c*, *ym*)

Select which channel(s) to display (c) and the Y scaling mode (ym). c may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2', 'both', 'chboth' ym may be: 'linear', '10dB', '2dB' (per division).

**set\_amp\_ref\_level\_db**(*self*, *r*)

Set the display full scale point to be *r* dB below input full scale. *r* will be forced to be a multiple of 10dB.

**get\_amp\_ref\_level\_db**(*self*)

Get the display full scale point in dB.

**window**(*self*, *w*)

Set the window function to *w*. *w* may be: 'flattop', 'hanning', 'uniform'.

**start\_measurement**(*self*, *a*, *c*, *m*)

Start a measurement. The averaging mode is *a*: 'off', 'rms'. The number of spectra to average is *c*. This will be made a power of 2 internally. The maximum time to wait for completion is *m* seconds (rounded up to a multiple of 5). N.B. Call this BEFORE `get_spectrum()`.

**read\_annotation**(*self*)

Read the display annotation.

**parse\_channel\_annotation**(*self*, *ann*, *c*)

Return the annotation for channel *c*. *ann* is the result of `read_annotation()`. *c* may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2'

**parse\_average\_annotation**(*self*, *ann*)

Return information on averaging from annotation *ann*.

**parse\_bandwidth\_annotation**(*self*, *ann*)

Return bandwidth information from annotation *ann*.

**get\_spectrum**(*self*)

Read a spectrum. This returns a tuple: (*freq*,*cha*,*chb*,*samps*,*aok*,*bok*). *freq* is a NumPy array of frequencies. *cha* is a NumPy array of channel A amplitudes at the *freq* frequencies. *chb* is a NumPy array of channel B amplitudes at the *freq* frequencies. *samps* is the number of values in the arrays. *aok* is True if channel A was measured. *bok* is True if channel B was measured.

**Inherited from GPIBWrap.GPIBInst(Section 1.3)**

`reset_bus()`, `set_display_status()`, `set_dump_gpib()`, `set_obj()`, `set_print_error()`,

set\_print\_gpib()

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

### 1.7.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 1.8 Class TEK\_710A



Sony/Tektronix RTD-710A Transient Digitizer Object.

### 1.8.1 Methods

<p><b>__init__</b>(<i>self</i>, <i>a</i>, <i>g</i>)</p> <p>x.__init__(...) initializes x; see help(type(x)) for signature</p> <p>Overrides: object.__init__ exitit(inherited documentation)</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><b>range</b>(<i>self</i>, <i>c</i>, <i>v</i>)</p> <hr/> <p>Set the sensitivity of channel <i>c</i> to <i>v</i> volts. <i>c</i> may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2'</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><b>offset</b>(<i>self</i>, <i>c</i>, <i>v</i>)</p> <hr/> <p>Set the offset of channel <i>c</i> to <i>v</i> volts. <i>c</i> may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2'</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**coupling**(*self*, *c*, *m*)

Set the coupling mode for channel *c*. *c* may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2'  
*m* may be: 'ac', 'dc', 'gnd', 'ac\_lf\_reject', 'ac\_hf\_reject'

**bw\_limit**(*self*, *b*)

Turn on bandwidth limiting (*b*=1) or turn it off (*b*=0).

**ch1\_only**(*self*, *b*)

Turn on Channel 1 only mode (*b*=1) or turn it off (*b*=0).

**sample\_interval**(*self*, *t*)

Set the interval between samples to be *t* seconds.

**samples**(*self*, *n*)

Set the number of samples to take to *n*.

**sample\_hold**(*self*, *b*)

Stop sampling (*b*=1) or start again (*b*=0).

**record\_wave**(*self*, *waitms*)

Record a waveform. Allow *waitms* milliseconds for this to complete.

**read\_wave**(*self*, *location*, *c*)

Read a waveform back from a specified location number and channel, *c*. *chan* may be: 'a', 'cha', 'b', 'chb', 'ch1', 'ch2'

### ***Inherited from GPIBWrap.GPIBInst(Section 1.3)***

reset\_bus(), set\_display\_status(), set\_dump\_gpib(), set\_obj(), set\_print\_error(),  
 set\_print\_gpib()

### ***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

## **1.8.2 Properties**

---

Name	Description
<i>Inherited from object</i> __class__	

## Index

- GPIBWrap (*module*), 2–13
  - GPIBWrap.ampunits (*function*), 2
  - GPIBWrap.argerr (*function*), 2
  - GPIBWrap.callerr (*function*), 2
  - GPIBWrap.chancodes (*function*), 2
  - GPIBWrap.couplecodes (*function*), 2
  - GPIBWrap.eu (*function*), 2
  - GPIBWrap.frequnits (*function*), 2
  - GPIBWrap.GPIBInst (*class*), 2–3
    - GPIBWrap.GPIBInst.reset\_bus (*method*), 3
    - GPIBWrap.GPIBInst.set\_display\_status (*method*), 3
    - GPIBWrap.GPIBInst.set\_dump\_gpib (*method*), 3
    - GPIBWrap.GPIBInst.set\_obj (*method*), 3
    - GPIBWrap.GPIBInst.set\_print\_error (*method*), 3
    - GPIBWrap.GPIBInst.set\_print\_gpib (*method*), 3
  - GPIBWrap.HP\_3325A (*class*), 5–7
    - GPIBWrap.HP\_3325A.check\_error (*method*), 5
    - GPIBWrap.HP\_3325A.check\_open (*method*), 5
    - GPIBWrap.HP\_3325A.set\_amplitude (*method*), 6
    - GPIBWrap.HP\_3325A.set\_dc\_offset (*method*), 6
    - GPIBWrap.HP\_3325A.set\_frequency (*method*), 5
    - GPIBWrap.HP\_3325A.set\_function (*method*), 5
    - GPIBWrap.HP\_3325A.set\_phase (*method*), 6
    - GPIBWrap.HP\_3325A.set\_sweep\_end\_frequency (*method*), 6
    - GPIBWrap.HP\_3325A.set\_sweep\_marker\_frequency (*method*), 6
    - GPIBWrap.HP\_3325A.set\_sweep\_mode (*method*), 6
    - GPIBWrap.HP\_3325A.set\_sweep\_start\_frequency (*method*), 6
    - GPIBWrap.HP\_3325A.set\_sweep\_time (*method*), 6
    - GPIBWrap.HP\_3325A.start\_sweep\_continuous (*method*), 6
    - GPIBWrap.HP\_3325A.start\_sweep\_single (*method*), 6
    - GPIBWrap.HP\_3325A.use\_amplitude\_modulation (*method*), 6
    - GPIBWrap.HP\_3325A.use\_phase\_modulation (*method*), 7
  - GPIBWrap.HP\_3457A (*class*), 3–5
    - GPIBWrap.HP\_3457A.check\_open (*method*), 4
    - GPIBWrap.HP\_3457A.clear\_fast\_mode (*method*), 4
    - GPIBWrap.HP\_3457A.read (*method*), 4
    - GPIBWrap.HP\_3457A.set\_fast\_mode (*method*), 4
    - GPIBWrap.HP\_3457A.set\_function (*method*), 4
  - GPIBWrap.HP\_3582A (*class*), 8–11
    - GPIBWrap.HP\_3582A.check\_open (*method*), 9
    - GPIBWrap.HP\_3582A.coupling (*method*), 9
    - GPIBWrap.HP\_3582A.display (*method*), 9
    - GPIBWrap.HP\_3582A.get\_amp\_ref\_level\_db (*method*), 10
    - GPIBWrap.HP\_3582A.get\_spectrum (*method*), 10
    - GPIBWrap.HP\_3582A.parse\_average\_annotation (*method*), 10
    - GPIBWrap.HP\_3582A.parse\_bandwidth\_annotation (*method*), 10
    - GPIBWrap.HP\_3582A.parse\_channel\_annotation (*method*), 10
    - GPIBWrap.HP\_3582A.read\_annotation (*method*), 10

- GPIBWrap.HP\_3582A.set\_amp\_ref\_level\_db  
(*method*), 9
- GPIBWrap.HP\_3582A.set\_frequency (*method*),  
9
- GPIBWrap.HP\_3582A.set\_input\_mode  
(*method*), 9
- GPIBWrap.HP\_3582A.set\_sensitivity (*method*),  
9
- GPIBWrap.HP\_3582A.set\_sensitivity\_db  
(*method*), 9
- GPIBWrap.HP\_3582A.start\_measurement  
(*method*), 10
- GPIBWrap.HP\_3582A.window (*method*),  
10
- GPIBWrap.HP\_8903E (*class*), 7–8
  - GPIBWrap.HP\_8903E.check\_open (*method*),  
7
  - GPIBWrap.HP\_8903E.get\_ac\_volts (*method*),  
8
  - GPIBWrap.HP\_8903E.get\_distortion (*method*),  
8
- GPIBWrap.TEK\_710A (*class*), 11–13
  - GPIBWrap.TEK\_710A.bw\_limit (*method*),  
12
  - GPIBWrap.TEK\_710A.ch1\_only (*method*),  
12
  - GPIBWrap.TEK\_710A.coupling (*method*),  
11
  - GPIBWrap.TEK\_710A.offset (*method*),  
11
  - GPIBWrap.TEK\_710A.range (*method*),  
11
  - GPIBWrap.TEK\_710A.read\_wave (*method*),  
12
  - GPIBWrap.TEK\_710A.record\_wave (*method*),  
12
  - GPIBWrap.TEK\_710A.sample\_hold (*method*),  
12
  - GPIBWrap.TEK\_710A.sample\_interval  
(*method*), 12
  - GPIBWrap.TEK\_710A.samples (*method*),  
12