agilentlightwave Documentation

Release 0.2

Jean-Luc Tambasco

Jan 15, 2019

Contents

Contents:

CHAPTER 1

Introduction

Documentation for the Agilent Lightwave Python 3 driver.

CHAPTER 2

Driver

2.1 Lightwave system

2.1.1 Classes

AgilentLightwave(gpib_num, gpib_dev_num, ...)	Agilent Lightwave driver object including power meter and laser.
AgilentLightwaveConnection(gpib_num, ...)	
LaserAgilent8164B([gpib_num, gpib_dev_num, ...])	Controls the laser module in the Agilent 8164B.
PowerMeterAgilent8164B([gpib_num, ...])	Driver for the Agilent 8164B power meter module that can read power and set the wavelength as well as specify in what units the Agilent 8164B should operate in.

AgilentLightwave

class AgilentLightwave(*gpib_num*, *gpib_dev_num*, *pm_sensor_num*, *pm_channel_num=1*, *laser_output_mode='high'*, *power_unit='W'*)

Bases: object

Agilent Lightwave driver object including power meter and laser.

Parameters

- **gpib_num** (*int*) – The number of the GPIB bus the power meter is sitting on.
- **gpib_dev_num** (*int*) – The device number that the power meter is on the aforementioned bus.
- **pm_sensor_num** (*int*) – The slot containing the desired power meter.
- **pm_channel_num** (*int*) – Either *1* or *2* depending on which power metre channel to use. Some power meters only have one channel so this should then be set to *1*.

- **laser_output_mode** (*str*) – 'HIGH' -> The High Power output is regulated. 'LOWS' -> The Low SSE output is regulated. 'BHR' -> Both outputs are active but only the

  High Power output is Regulated.

  'BLR' -> Both outputs are active but only the Low SSE output is Regulated.

- **power_unit** (*str*) – Either 'W' or 'dBm' depending on whether the power units should be displayed in [W] or [dBm] on the Agilent 8164B's screen.

system
_AgilentLightwaveSystem – Contains system level functions.

laser
agilent_8164B_laser – Contains laser functions.

power_meter
agilent_8164B_power_meter – Contains power_meter functions.

2.1.2 Class Inheritance Diagram

2.2 Laser

2.2.1 Classes

`AgilentLightwaveConnection`(gpib_num, ...)	
LaserAgilent8164B([gpib_num, gpib_dev_num, ...])	Controls the laser module in the Agilent 8164B.

LaserAgilent8164B

class LaserAgilent8164B(*gpib_num=None*, *gpib_dev_num=None*, *power_unit='W'*, *output_mode='high'*)

Bases: `agilentlightwave._agilent_lightwave_connection.AgilentLightwaveConnection`, `agilentlightwave._laser.LaserTunable`

Controls the laser module in the Agilent 8164B.

Parameters

- **gpib_num** (*int*) – The GPIB bus number the laser is on.
- **gpib_dev_num** (*int*) – The device number the laser in on the bus.

- **power_unit** (*str*) – Either 'W' or 'dBm' depending on what units the Agilent 8164B should use for the laser power.
- **output_mode** (*str*) – 'HIGH' -> The High Power output is regulated. 'LOWS' -> The Low SSE output is regulated. 'BHR' -> Both outputs are active but only the High Power output is Regulated. 'BLR' -> Both outputs are active but only the Low SSE output is Regulated.

Methods Summary

dbm_to_watts(power_dbm)	Converts [dBm] to [W].
get_on_or_off()	Checks if the laser is on or off.
get_power_W()	Gets the power of the laser in [W].
get_power_dbm()	
get_power_mW()	
get_power_nW()	
get_power_pW()	
get_power_uW()	
get_unit()	Gets the units the laser is operating in.
get_velocity_nm_s()	
get_wavelength_m()	Gets the wavelength in [m] of the laser.
get_wavelength_mm()	
get_wavelength_nm()	
get_wavelength_power_scan_manual(...[, ...])	Performs a wavelength sweep.
get_wavelength_um()	
last_operation_completed()	
set_power_W(power_W)	Sets the power of the laser in [W].
set_power_dbm(power_dbm)	
set_power_mW(power_mW)	
set_power_uW(power_uW)	
set_unit(unit)	Sets the units the laser should operate in.
set_velocity_nm_s(sweep_speed_nm_per_sec)	
set_wavelength_m(wavelength_m)	Sets the wavelength in [m] of the laser.
set_wavelength_nm(wavelength)	
set_wavelength_um(wavelength)	
turn_off()	Turns the laser off.
turn_on()	Turns the laser on.
wait_for_last_operation_completed()	
watts_to_dbm(power_W)	Converts [W] to [dBm].
wavelength_sweep(start_wavelength_nm, ...[, ...])	
wavelength_sweep_interp(start_wavelength_nm, ...)	
wavelength_sweep_manual_interp(...[, ...])	

Methods Documentation

static dbm_to_watts(*power_dbm*)
Converts [dBm] to [W].

Parameters **power_dbm** (*int, float*) – Power in [dBm].

Returns Power in [W].

Return type float

get_on_or_off()
Checks if the laser is on or off.

Returns *True* if the laser is off, *False* if the laser is on.

Return type bool

get_power_W()
Gets the power of the laser in [W].

Returns The power of the laser in [W].

Return type float

get_power_dbm()

get_power_mW()

get_power_nW()

get_power_pW()

get_power_uW()

get_unit()
Gets the units the laser is operating in.

The units affect in what units the laser power is displayed on the screen.

Returns The units the laser is oeprating in.

Return type str

get_velocity_nm_s()

get_wavelength_m()
Gets the wavelength in [m] of the laser.

Returns The wavelength in metres the laser is set to.

Return type float

get_wavelength_mm()

get_wavelength_nm()

get_wavelength_power_scan_manual(*start_wavelength_nm*, *stop_wavelength_nm*, *step_wavelength_nm*, *power_meters=None*, *delay_wavelength_changes_s=1.0*, *filename=None*)
Performs a wavelength sweep.

Manually changes the laser power and the measures. Does not use the in-built sweep function like Thach's sweep does.

Parameters

- **start_wavelength_nm** (*int, float*) – The minimum wavelength value to start the sweep.
- **stop_wavelength_nm** (*int, float*) – The maximum wavelength value to stop the sweep.
- **step_wavelength_nm** (*int, float*) – The wavelength stepsize.

- **power_meters** (*PowerMeter, list, None*) – The constructed power meter object that should be read as the wavelength is swept. If a list of *PowerMeter* is given, all power meters in the list will be read. *None* if no power meter should be read.
- **power_meter_units** (*str*) – 'W' if the power should be read and returned in Watts, or 'dBm' if the power should be read and returned in 'dBm'.
- **delay_wavelength_changes_s** (*int, float*) – The delay in seconds between laser wavelength changes.
- **filename** (*None, str*) – The name of the file to save the wavelength and power data to. The data is not saved to a file if *None*.

Returns

The first item is a float list of the wavelengths swept. Subsequent items in the list are power meter readings given in the same order as in *power_meters.*

Return type list

get_wavelength_um()

last_operation_completed()

set_power_W(*power_W*)
Sets the power of the laser in [W].

Parameters **power_W** (*int, float*) – power in [W] to set the laser to.

Returns The power of the laser.

Return type float

set_power_dbm(*power_dbm*)

set_power_mW(*power_mW*)

set_power_uW(*power_uW*)

set_unit(*unit*)
Sets the units the laser should operate in.

This will affect in what units the laser power is displayed on the screen.

Parameters **unit** (*str*) – Either 'dBm' or 'W'.

Returns The units the laser is oeprating in.

Return type str

set_velocity_nm_s(*sweep_speed_nm_per_sec*)

set_wavelength_m(*wavelength_m*)
Sets the wavelength in [m] of the laser.

Parameters **wavelength_m** (*int, float*) – The wavelength in metres to set the laser to.

Returns The wavelength in metres the laser was set to.

Return type float

set_wavelength_nm(*wavelength*)

set_wavelength_um(*wavelength*)

turn_off()
Turns the laser off.

Returns

***True* if the laser is off, *False* if the laser is on.** (Should be *False*.)

Return type bool

turn_on()
Turns the laser on.

Returns

***True* if the laser is on, *False* if the laser is off.** (Should be *True*.)

Return type bool

wait_for_last_operation_completed()

static watts_to_dbm(*power_W*)
Converts [W] to [dBm].

Parameters **power_W** (*int, float*) – Power in [W].

Returns Power in [dBm].

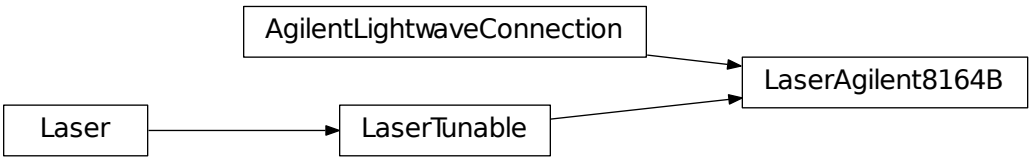
Return type float

wavelength_sweep(*start_wavelength_nm*, *stop_wavelength_nm*, *step_wavelength_nm*, *max_power_mW*, *sweep_speed_nm_per_sec=5.0*, *power_metre_slot=1*, *int_time_ms=20.0*, *filename=None*, *preserve_settings=True*)

wavelength_sweep_interp(*start_wavelength_nm*, *stop_wavelength_nm*, *step_wavelength_nm*, *max_power_W*, *sweep_speed_nm_per_sec=5.0*, *filename=None*)

wavelength_sweep_manual_interp(*start_wavelength_nm*, *stop_wavelength_nm*, *step_wavelength_nm*, *power_meters*, *delay_wavelength_changes_s=1.0*, *filename=None*)

2.2.2 Class Inheritance Diagram



2.3 Power meter

2.3.1 Classes

AgilentLightwaveConnection(gpib_num, ...)

Continued on next page

Table 4 – continued from previous page

PowerMeterAgilent8164B([gpib_num, ...])	Driver for the Agilent 8164B power meter module that can read power and set the wavelength as well as specify in what units the Agilent 8164B should operate in.

PowerMeterAgilent8164B

class PowerMeterAgilent8164B(*gpib_num=None*, *gpib_dev_num=None*, *sensor_num=1*, *channel_num=1*, *power_unit='W'*)

Bases: agilentlightwave._agilent_lightwave_connection.AgilentLightwaveConnection, agilentlightwave._power_meter.PowerMeter

Driver for the Agilent 8164B power meter module that can read power and set the wavelength as well as specify in what units the Agilent 8164B should operate in.

Parameters

- **gpib_num** (*int*) – The number of the GPIB bus the power meter is sitting on.
- **gpib_dev_num** (*int*) – The device number that the power meter is on the aforementioned bus.
- **channel_num** (*int*) – Either *1* or *2* depending on which power metre channel to use.
- **power_unit** (*str*) – Either 'W' or 'dBm' depending on whether the power units should be displayed in [W] or [dBm] on the Agielent 8164B's screen.

Methods Summary

dbm_to_watts(power_dbm)	Converts [dBm] to [W].
get_analogue_current_A(analogue_voltage_V)	Get the detector photocurrent [A].
get_analogue_power_W(analogue_voltage_V)	Calculates the power of the power meter based on the analogue voltage value read.
get_auto_range()	Checks what the auto-range is set to.
get_averaging_time_s()	
get_power_W([average, read_period_ms])	Read the power meter.
get_power_dbm([average, read_period_ms])	
get_power_mW([average, read_period_ms])	
get_power_meter_range()	
get_power_nW([average, read_period_ms])	
get_power_pW([average, read_period_ms])	
get_power_uW([average, read_period_ms])	
get_range_dbm()	
get_responsivity_A_W()	Returns the responsivity [A/W] of the detector.
get_unit()	Gets the units the power meter is set to display.
get_wavelength_m()	Gets the wavelength the power meter is set to read.
get_wavelength_mm()	
get_wavelength_nm()	
get_wavelength_um()	
set_auto_range()	Turns on auto-range.
set_averaging_time_s(averaging_time_s)	
set_power_meter_range(max_power_dbm)	Set the power meter range, range in dBm
set_range_dbm(range_dbm)	

Continued on next page

Table 5 – continued from previous page

set_unit(unit)	Sets the units the power meter will display on screen.
set_wavelength_m(wavelength_m)	Sets the wavelength the power meter is set to read.
set_wavelength_nm(wavelength_nm)	
set_wavelength_um(wavelength_um)	
unset_auto_range()	Turns off auto-range.
watts_to_dbm(power_W)	Converts [W] to [dBm].

Methods Documentation

static dbm_to_watts(*power_dbm*)
Converts [dBm] to [W].

Parameters **power_dbm** (*int, float*) – Power in [dBm].

Returns Power in [W].

Return type float

get_analogue_current_A(*analogue_voltage_V*)
Get the detector photocurrent [A]. This should be directly propertional to the detector analogue voltage [V] output.

Parameters **analogue_voltage_V** (*float, np.array*) – Analogue voltage [V] representing the current.

Returns Analogue current [A] corresponding to the given analogue voltage [V].

Return type float, np.array

get_analogue_power_W(*analogue_voltage_V*)
Calculates the power of the power meter based on the analogue voltage value read. The analogue voltage should be directly related to the current.

Parameters **analogue_voltage_V** (*float, np.array*) – Analogue voltage [V] representing the current.

Returns

The power [W] corresponding to the given analogue voltage [V].

Return type float, np.array

get_auto_range()
Checks what the auto-range is set to.

Returns

***1* if auto-range is switched on and** *0* if auto-range is switched off.

Return type bool

get_averaging_time_s()

get_power_W(*average=1*, *read_period_ms=250.0*)
Read the power meter.

Parameters

- **average** (*int*) – How many power readings should be averaged over before returning a power.
- **read_period_ms** (*int, float*) – The time to wait between power readings. Only makes sense if *average* > *1*.

Returns The, perhaps averaged, power reading.

Return type float

get_power_dbm(*average=1*, *read_period_ms=None*)

get_power_mW(*average=1*, *read_period_ms=None*)

get_power_meter_range()

get_power_nW(*average=1*, *read_period_ms=None*)

get_power_pW(*average=1*, *read_period_ms=None*)

get_power_uW(*average=1*, *read_period_ms=None*)

get_range_dbm()

get_responsivity_A_W()
Returns the responsivity [A/W] of the detector.

Returns Responsivity [A/W] of the detector.

Return type float

get_unit()
Gets the units the power meter is set to display.

Returns

The unit the power meter is set to display. Either 'W' or 'dBm'

Return type str

get_wavelength_m()
Gets the wavelength the power meter is set to read.

Returns The wavelength in [m] the power meter is operating in.

Return type float

get_wavelength_mm()

get_wavelength_nm()

get_wavelength_um()

set_auto_range()
Turns on auto-range.

Returns

***1* if auto-range is switched on and** *0* if auto-range is switched off.

Return type bool

set_averaging_time_s(*averaging_time_s*)

set_power_meter_range(*max_power_dbm*)
Set the power meter range, range in dBm

set_range_dbm(*range_dbm*)

set_unit(*unit*)
Sets the units the power meter will display on screen.

Parameters **unit** (*str*) – Either 'W' or 'dBm' depending on whether the power units should be displayed in [W] or [dBm] on the Agielent 8164B's screen.

Returns

The unit the power meter is set to display. Either 'W' or 'dBm'

Return type str

set_wavelength_m(*wavelength_m*)
Sets the wavelength the power meter is set to read.

Parameters **wavelength_m** (*int, float*) – The wavelength in [m] to set the power meter to.

Returns

The wavelength the power meter is operating in.

Return type float

set_wavelength_nm(*wavelength_nm*)

set_wavelength_um(*wavelength_um*)

unset_auto_range()
Turns off auto-range.

Returns

***1* if auto-range is switched on and** *0* if auto-range is switched off.

Return type bool

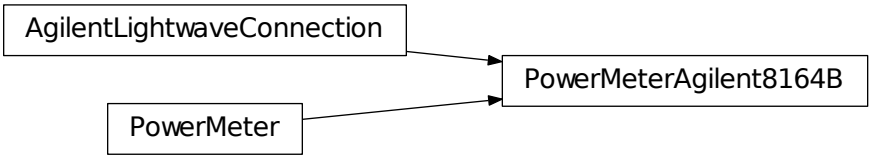
static watts_to_dbm(*power_W*)
Converts [W] to [dBm].

Parameters **power_W** (*int, float*) – Power in [W].

Returns Power in [dBm].

Return type float

2.3.2 Class Inheritance Diagram



Python Module Index

a

Index

T

U

W