
Owon VDS6104 Oscilloscope Drivers

Release 0.1-testing

David Standiford

Jul 14, 2021

CONTENTS:

1	OwonVDS6104	1
1.1	offsets module	1
1.2	owon_vds6104 module	2
1.3	scale module	7
1.4	support module	9
1.5	test module	9
1.6	test_owon_vds6104 module	9
2	Indices and tables	11
	Python Module Index	13
	Index	15

OWONVDS6104

1.1 offsets module

```
class offsets.Bunch(d)
    Bases: dict

class offsets.dotdict
    Bases: dict
    dot.notation access to dictionary attributes

class offsets.offset
    Bases: object
    Offsets for OwonVDS6104 oscilloscope

    ADC_AVG_CH1 = 88
    ADC_AVG_CH2 = 90
    ADC_AVG_CH3 = 92
    ADC_AVG_CH4 = 94
    ADC_MAX_CH1 = 80
    ADC_MAX_CH2 = 82
    ADC_MAX_CH3 = 84
    ADC_MAX_CH4 = 86
    ADC_MIN_CH1 = 72
    ADC_MIN_CH2 = 74
    ADC_MIN_CH3 = 76
    ADC_MIN_CH4 = 78
    ADC_OVER_FLAG = 70
    ADC_PRECISION = 14
    CHECK = 4
    CH_NUMS = 16
    DRAW_MODE = 30
    DYNAMIC_CHECK = 8
```

```
FFT_SIZE = 26
FFT_VALID = 24
FRAME_NUMS = 22
FREQ_CH1 = 38
FREQ_CH2 = 42
FREQ_CH3 = 46
FREQ_CH4 = 50
INFO_SIZE = 10
REF_FREQ_CH1 = 54
REF_FREQ_CH2 = 58
REF_FREQ_CH3 = 62
REF_FREQ_CH4 = 66
ROLL_DATA_SIZE = 34
ROLL_MODE = 32
RUN_STATUS = 12
START = 0
TRIG_TYPE = 96
VOLTSCALE_CH1 = 260
VOLTSCALE_CH2 = 262
VOLTSCALE_CH3 = 264
VOLTSCALE_CH4 = 266
WAVE_DATA_SIZE = 18
ZERO_CH1 = 268
ZERO_CH2 = 272
ZERO_CH3 = 276
ZERO_CH4 = 280
```

```
offsets.to_bunch(d)
```

1.2 owon_vds6104 module

```
class owon_vds6104.OwonVDS6104(address)
    Bases: object
    property acquire_mode
    autoset()
        Perform autoset
    capture(channel)
```

check_measurement_overflow()

Check for ADC overflow condition

check_model()

Verify the model of the selected instrument matches what is expected.

The format of the data returned from the instrument should be in the format of: "OWON <model no.> <serial number> VX.XX.XX"

Parameters **bool** – True:Correct Model, False:Invalid model

force_trig()

Command device to force an acquisition trigger

get_autoset_progress()

Gets autoset progress :return int: [1-100] percent

get_bw_limit(channel)

Get bandwidth limit of channel

Parameters **channel** (*int*) – [1,2,3,4] Channel to get

Return **bool** True if enabled, False otherwise

get_calibration_progress()

Get self-calibration progress :return int:[1-100] percent

get_channel_state(channel)

Get the display state of channel

Parameters **channel** (*int*) – Channel selection [1,2,3,4]

Return **bool** State of channel display

get_coupling(channel)**get_scale(channel) → float**

Get the current vertical division scale for channel

Parameters **int** – channel: [1,2,3,4] Channel selection

Return **float** Voltage of selected channel scale

get_trig_status()

Get the current trigger status

Return **string** "AUTO", "STOP", "SCAN" or "TRIG"

get_vertical_offset(channel) → float

Get vertical offset of the channel

Parameters **channel** (*int*) – Channel number [1,2,3,4]

Return **float** offset, in divisions

Note voltage can be calculated by offset * scale_voltage

ident()

Query instrument for identification

Return **string** Identity from instrument

measure(channel, function)

measurement function that supports many different hardware measurements.

Parameters

- **channel** (*int*) – [1,2,3,4] Channel to perform measurement on
- **function** (*enum*) –
 measurement.<function> where <function> can be:
 - vmax - maximum value, volts
 - vmin - minimum value, volts
 - vpp - peak to peak, volts
 - vtop - top value, volts
 - vamp - amplitude value, volts
 - vavg - average value, volts
 - vrms - rms value, volts
 - crms - cycle rms value, volts
 - overshoot - overshoot, percent
 - preshoot - preshoot, percent
 - pos_duty - positive duty cycle, percent
 - neg_duty - negative duty cycle, percent
 - period - cycle time, seconds
 - frequency - frequency, hertz
 - rise_time - rise time, seconds
 - fall_time - fall time, seconds
 - pos_width - positive pulse width time, seconds
 - neg_width - negative pulse width time, seconds
 - area - area, volt-seconds
 - cyc_area - cycle area, volt-seconds
 - pos_puls - positive pulse count, integer
 - neg_puls - negative pulse count, integer
 - ris_edg_cnt - number of rising edges, integer
 - neg_edg_cnt - number of falling edges, number

Return string string representation of measurement

property measurement_source

property memory_depth

Get current memory depth of device

property precision

query(*command*)

Send query to instrument, strip newlines

run()

Start running device

scale_init()

self_calibrate()

Perform self calibration

send(*command*)

set_bw_limit(*channel, mode*)

Set bandwidth limit. Supports only 20MHz limit.

Parameters

- **channel** (*int*) – [1,2,3,4] Channel of limit
- **mode** (*enum*) – [state.enable, state.disable]

Example >set_bw_limit(1, state.enable)

set_channel_state(*channel, mode*)

Configure display state of channels

Parameters

- **channel** (*int*) – [1,2,3,4] Channel selection
- **mode** (*int*) – state.enable or state.disable

set_coupling(*channel, mode*)

Set the oscilloscope channel coupling mode

Parameters

- **channel** (*int*) – channel number [1,2,3,4]
- **coupling** (*string*) – coupling mode ['ac', 'dc', 'gnd']

Note Enum coupling.[mode] also supported

set_scale(*channel, voltage*)

Set the vertical division scale

Parameters

- **channel** (*int*) –
- **scale** (*float*) –

set_vertical_offset(*channel, offset*)

Set vertical offset for input channel

Parameters **channel** (*int*) – Channel number [1,2,3,4]

Note Range of allowable values varies with scale: - 2mV: -1000 to 1000 - 5mV: -400 to 400 - 10mV: -200 to 200 - 20mV: -100 to 100 - 50mV: -40 to 40 - 100mV: -200 to 200 - 500mV: -40 to 40 - 1V: -40 to 40 - 2V: -20 to 20 - 5V: -8 to 8

property state_measurement

stop()

Stop running device

property time_measurement: float

Gate interval of signal measurement function

Return float gate time interval

property timebase

```
property timebase_offset
trig_set_half()
    Set the trigger level to vertical mid-point
verbose(text)
vertical_init()
class owon_vds6104.acquire(value)
    Bases: enum.Enum
    An enumeration.
    peak = 'PEAK'
    sample = 'SAMP'
class owon_vds6104.coupling(value)
    Bases: enum.Enum
    Enum for channel coupling selection
    AC = 'ac'
    DC = 'dc'
    GND = 'gnd'
    ac = 'ac'
    dc = 'dc'
    gnd = 'gnd'
class owon_vds6104.measurement(value)
    Bases: enum.Enum
    Enum for measurement function selection
    area = 'AREA'
    crms = 'CRMS'
    cyc_area = 'CAR'
    fal_edg_cnt = 'FEDG'
    fall_time = 'FTIM'
    freq = 'FREQ'
    neg_duty = 'NDUT'
    neg_puls = 'NPUL'
    neg_width = 'NWID'
    overshoot = 'OVER'
    period = 'PER'
    pos_duty = 'PDUT'
    pos_puls = 'PPUL'
    pos_width = 'PWID'
    preshoot = 'PRES'
```

```

ris_edg_cnt = 'REDG'
rise_time = 'RTIM'
vamp = 'VAMP'
vavg = 'VAVG'
vbase = 'VBASE'
vmax = 'VMAX'
vmin = 'VMIN'
vpp = 'VPP'
vrms = 'VRMS'
vtop = 'VTOP'

class owon_vds6104.state(value)
    Bases: enum.Enum
    Enum for state selection
    disable = 0
    enable = 1

```

1.3 scale module

`scale.calc_valid_inputs(MIN_VAL, MAX_VAL, MANTISSA)`

Produce a list of possible values between

MIN_VAL and MAX_VAL using the MANTISSA exponent

Parameters

- **MIN_VAL** (*float*) – Minimum value to start with
- **MAX_VAL** (*float*) – Maximum value to end with
- **MANTISSA** (*list*) – Exponent values to use

Return list List of possible values

Example

```

>>> MIN_VAL = 2E-9
>>> MAX_VAL = 10
>>> MANTISSA = (1, 2, 5)
>>> calc_valid_inputs(MIN_VAL, MAX_VAL, MANTISSA)

```

[2e-09, 5e-09, 1e-08, 2e-08, 5e-08, 1e-07, 2e-07, 5e-07, 1e-06, 2e-06, 5e-06, 1e-05, 2e-05, 5e-05, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0]

Note: In math the Mantissa is the fractional part of the common base10 logarithm. Idea from Philipp Klaus/DS1054Z drivers.

`scale.check_index(target, dataset)`

Get the index of an input within dataset

Parameters

- **target** (*float*) – input to search for
- **dataset** (*list*) – sequence input to search in

Return int index where element exists

Example

```
>>> allowable_values = [1, 2, 5, 10, 20]
>>> check_index(10, allowable_values)
3
```

Note: This function does not distinguish between multiple occurrences of the search term. It will return the first occurrence only.

scale.get_closest_value(*input_value, MIN, MAX, MANTISSA*)

scale.get_val_ceil(*target, validated, vals_valid*)

Get the next wholly-encompassing value of the input within the valid input list.

Parameters

- **target** (*float*) – input command for comparison
- **validated** (*float*) – validated input for comparison
- **vals_valid** (*list*) – sequence of valid data

Return float next wholly-encompassing value up to the maximum within the allowable values

Example

```
>>> allowable_values = [1, 2, 5, 10, 20, 50]
>>> target = 10.1
>>> validated = 10
>>> get_val_ceil(target, validated, allowable_values)
20
```

scale.validate_input(*target, vals_valid*)

Input validation to find the lowest delta between input value and a list of valid inputs.

Parameters

- **target** (*float*) – data input for comparison
- **vals_valid** (*list*) – list of valid inputs

Return float lowest delta that exists in vals_valid

Example

```
>>> allowable_values = [1, 2, 5, 10, 20]
>>> validate_input(10.1, allowable_values)
10
```

1.4 support module

`support.convert_list_to_int(list_slice)`

Convert a list of bytes to a different data type. Currently supports return of 32 and 16 bit values

Parameters `list_slice (list)` – list of 1-byte values

Return int converted datatype

`support.get_available_instruments()`

List of available instruments found as VISA resources

Return list Available instrument IDs

`support.invert_endian(list_slice)`

Reverses the endian of a list

`support.plot_x_data(x, y)`

`support.u8_to_u32(list_slice)`

Convert U8 (unsigned char) to U32 datatype

1.5 test module

`test.validate_time(input_time)`

The time within the scope sometimes needs a decimal even though it is all zeros.

This function provides the proper validated numbers and zero pads based on contents of time_dec.

Parameters `input_time (float)` – Input time figure

Return string string representation of time

1.6 test_owon_vds6104 module

`class test_owon_vds6104.Test_OwonVDS6104(methodName='runTest')`

Bases: `unittest.case.TestCase`

`channels = [1, 2, 3, 4]`

`cmd_delay = 0.1`

`test_channel_dispay()`

” Test the display functionality for all channels

`timebases = [1e-09, 2e-09, 5e-09, 1e-08, 2e-08, 5e-08, 1e-07, 2e-07, 5e-07, 1e-06, 2e-06, 5e-06, 1e-05, 2e-05, 5e-05, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 100.0]`

`voltages = [0.002, 0.005, 0.01, 0.02, 0.05, 0.05, 0.1, 0.2, 0.5, 1, 2, 5]`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

O

offsets, [1](#)
owon_vds6104, [2](#)

S

scale, [7](#)
support, [9](#)

t

test, [9](#)
test_owon_vds6104, [9](#)

A

AC (*owon_vds6104.coupling attribute*), 6
ac (*owon_vds6104.coupling attribute*), 6
acquire (*class in owon_vds6104*), 6
acquire_mode (*owon_vds6104.OwonVDS6104 property*), 2
ADC_AVG_CH1 (*offsets.offset attribute*), 1
ADC_AVG_CH2 (*offsets.offset attribute*), 1
ADC_AVG_CH3 (*offsets.offset attribute*), 1
ADC_AVG_CH4 (*offsets.offset attribute*), 1
ADC_MAX_CH1 (*offsets.offset attribute*), 1
ADC_MAX_CH2 (*offsets.offset attribute*), 1
ADC_MAX_CH3 (*offsets.offset attribute*), 1
ADC_MAX_CH4 (*offsets.offset attribute*), 1
ADC_MIN_CH1 (*offsets.offset attribute*), 1
ADC_MIN_CH2 (*offsets.offset attribute*), 1
ADC_MIN_CH3 (*offsets.offset attribute*), 1
ADC_MIN_CH4 (*offsets.offset attribute*), 1
ADC_OVER_FLAG (*offsets.offset attribute*), 1
ADC_PRECISION (*offsets.offset attribute*), 1
area (*owon_vds6104.measurement attribute*), 6
autoset() (*owon_vds6104.OwonVDS6104 method*), 2

B

Bunch (*class in offsets*), 1

C

calc_valid_inputs() (*in module scale*), 7
capture() (*owon_vds6104.OwonVDS6104 method*), 2
CH_NUMS (*offsets.offset attribute*), 1
channels (*test_owon_vds6104.Test_OwonVDS6104 attribute*), 9
CHECK (*offsets.offset attribute*), 1
check_index() (*in module scale*), 7
check_measurement_overflow() (*owon_vds6104.OwonVDS6104 method*), 2
check_model() (*owon_vds6104.OwonVDS6104 method*), 3
cmd_delay (*test_owon_vds6104.Test_OwonVDS6104 attribute*), 9
convert_list_to_int() (*in module support*), 9

coupling (*class in owon_vds6104*), 6
crms (*owon_vds6104.measurement attribute*), 6
cyc_area (*owon_vds6104.measurement attribute*), 6

D

DC (*owon_vds6104.coupling attribute*), 6
dc (*owon_vds6104.coupling attribute*), 6
disable (*owon_vds6104.state attribute*), 7
dotdict (*class in offsets*), 1
DRAW_MODE (*offsets.offset attribute*), 1
DYNAMIC_CHECK (*offsets.offset attribute*), 1

E

enable (*owon_vds6104.state attribute*), 7

F

fal_edg_cnt (*owon_vds6104.measurement attribute*), 6
fall_time (*owon_vds6104.measurement attribute*), 6
FFT_SIZE (*offsets.offset attribute*), 1
FFT_VALID (*offsets.offset attribute*), 2
force_trig() (*owon_vds6104.OwonVDS6104 method*), 3
FRAME_NUMS (*offsets.offset attribute*), 2
freq (*owon_vds6104.measurement attribute*), 6
FREQ_CH1 (*offsets.offset attribute*), 2
FREQ_CH2 (*offsets.offset attribute*), 2
FREQ_CH3 (*offsets.offset attribute*), 2
FREQ_CH4 (*offsets.offset attribute*), 2

G

get_autoset_progress() (*owon_vds6104.OwonVDS6104 method*), 3
get_available_instruments() (*in module support*), 9
get_bw_limit() (*owon_vds6104.OwonVDS6104 method*), 3
get_calibration_progress() (*owon_vds6104.OwonVDS6104 method*), 3
get_channel_state() (*owon_vds6104.OwonVDS6104 method*), 3

`get_closest_value()` (in module scale), 8
`get_coupling()` (owon_vds6104.OwonVDS6104 method), 3
`get_scale()` (owon_vds6104.OwonVDS6104 method), 3
`get_trig_status()` (owon_vds6104.OwonVDS6104 method), 3
`get_val_ceil()` (in module scale), 8
`get_vertical_offset()` (owon_vds6104.OwonVDS6104 method), 3
`GND` (owon_vds6104.coupling attribute), 6
`gnd` (owon_vds6104.coupling attribute), 6

I

`ident()` (owon_vds6104.OwonVDS6104 method), 3
`INFO_SIZE` (offsets.offset attribute), 2
`invert_endian()` (in module support), 9

M

`measure()` (owon_vds6104.OwonVDS6104 method), 3
`measurement` (class in owon_vds6104), 6
`measurement_source` (owon_vds6104.OwonVDS6104 property), 4
`memory_depth` (owon_vds6104.OwonVDS6104 property), 4
`module`
 offsets, 1
 owon_vds6104, 2, 9
 scale, 7
 support, 9
 test, 9
 test_owon_vds6104, 9

N

`neg_duty` (owon_vds6104.measurement attribute), 6
`neg_puls` (owon_vds6104.measurement attribute), 6
`neg_width` (owon_vds6104.measurement attribute), 6

O

`offset` (class in offsets), 1
`offsets`
 module, 1
`overshoot` (owon_vds6104.measurement attribute), 6
`owon_vds6104`
 module, 2, 9
`OwonVDS6104` (class in owon_vds6104), 2

P

`peak` (owon_vds6104.acquire attribute), 6
`period` (owon_vds6104.measurement attribute), 6
`plot_x_data()` (in module support), 9
`pos_duty` (owon_vds6104.measurement attribute), 6

`pos_puls` (owon_vds6104.measurement attribute), 6
`pos_width` (owon_vds6104.measurement attribute), 6
`precision` (owon_vds6104.OwonVDS6104 property), 4
`preshoot` (owon_vds6104.measurement attribute), 6

Q

`query()` (owon_vds6104.OwonVDS6104 method), 4

R

`REF_FREQ_CH1` (offsets.offset attribute), 2
`REF_FREQ_CH2` (offsets.offset attribute), 2
`REF_FREQ_CH3` (offsets.offset attribute), 2
`REF_FREQ_CH4` (offsets.offset attribute), 2
`ris_edg_cnt` (owon_vds6104.measurement attribute), 6
`rise_time` (owon_vds6104.measurement attribute), 7
`ROLL_DATA_SIZE` (offsets.offset attribute), 2
`ROLL_MODE` (offsets.offset attribute), 2
`run()` (owon_vds6104.OwonVDS6104 method), 4
`RUN_STATUS` (offsets.offset attribute), 2

S

`sample` (owon_vds6104.acquire attribute), 6
`scale`
 module, 7
`scale_init()` (owon_vds6104.OwonVDS6104 method), 4
`self_calibrate()` (owon_vds6104.OwonVDS6104 method), 5
`send()` (owon_vds6104.OwonVDS6104 method), 5
`set_bw_limit()` (owon_vds6104.OwonVDS6104 method), 5
`set_channel_state()` (owon_vds6104.OwonVDS6104 method), 5
`set_coupling()` (owon_vds6104.OwonVDS6104 method), 5
`set_scale()` (owon_vds6104.OwonVDS6104 method), 5
`set_vertical_offset()` (owon_vds6104.OwonVDS6104 method), 5
`START` (offsets.offset attribute), 2
`state` (class in owon_vds6104), 7
`state_measurement` (owon_vds6104.OwonVDS6104 property), 5
`stop()` (owon_vds6104.OwonVDS6104 method), 5
`support`
 module, 9
`T`
`test`
 module, 9
`test_channel_dispay()` (test_owon_vds6104.Test_OwonVDS6104 method), 9

test_owon_vds6104
 module, 9
 Test_OwonVDS6104 (class in test_owon_vds6104), 9
 time_measurement (owon_vds6104.OwonVDS6104
 property), 5
 timebase (owon_vds6104.OwonVDS6104 property), 5
 timebase_offset (owon_vds6104.OwonVDS6104
 property), 5
 timebases (test_owon_vds6104.Test_OwonVDS6104 at-
 tribute), 9
 to_bunch() (in module offsets), 2
 trig_set_half() (owon_vds6104.OwonVDS6104
 method), 6
 TRIG_TYPE (offsets.offset attribute), 2

U

u8_to_u32() (in module support), 9

V

validate_input() (in module scale), 8
 validate_time() (in module test), 9
 vamp (owon_vds6104.measurement attribute), 7
 vavg (owon_vds6104.measurement attribute), 7
 vbase (owon_vds6104.measurement attribute), 7
 verbose() (owon_vds6104.OwonVDS6104 method), 6
 vertical_init() (owon_vds6104.OwonVDS6104
 method), 6
 vmax (owon_vds6104.measurement attribute), 7
 vmin (owon_vds6104.measurement attribute), 7
 voltages (test_owon_vds6104.Test_OwonVDS6104 at-
 tribute), 9
 VOLTSCALE_CH1 (offsets.offset attribute), 2
 VOLTSCALE_CH2 (offsets.offset attribute), 2
 VOLTSCALE_CH3 (offsets.offset attribute), 2
 VOLTSCALE_CH4 (offsets.offset attribute), 2
 vpp (owon_vds6104.measurement attribute), 7
 vrms (owon_vds6104.measurement attribute), 7
 vtop (owon_vds6104.measurement attribute), 7

W

WAVE_DATA_SIZE (offsets.offset attribute), 2

Z

ZERO_CH1 (offsets.offset attribute), 2
 ZERO_CH2 (offsets.offset attribute), 2
 ZERO_CH3 (offsets.offset attribute), 2
 ZERO_CH4 (offsets.offset attribute), 2