Owon VDS6104 Oscilloscope Drivers

Release 0.1-testing

David Standiford

Jul 14, 2021

CONTENTS:

CHAPTER

ONE

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 aquistion 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 occurances of the search term. It will return the first occurance 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]

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

I

M

N

O

P

Q

R

S

T

U

V

W

Z