**class MDACBasicChannel(InstrumentChannel):**

**Description:** Channel Class for MDAC

**Notes:**

Base Channel Class for MDAC DACs.

One instance is automatically created for each DAC in the MDAC chassis

Ie: mdac.chan01, mdac.chan02 .. mdac.chan64

One instance is also created for the internal ISENSE DAC

Ie: mdac.ch\_isense

**Example Usage**: N/A - When an MDAC object is created, MDACChannel(MBasicChannel) objects are automatically created for each DAC.

**Function:** \_\_init\_\_(self, parent, name, channum)

**Description:**

Object creation

Args:

parent (Instrument): The instrument the channel is a part of.

name (str): the name of the channel

channum (int): MDAC channel number ([1-64] for populated MDAC)

**Example Usage**: N/A

**Parameter:** voltage – R/W

Channel {x} voltage

**Units:** Volts

**Range:** -5 to 5

**Description**:

current output voltage [volts], set immediately

**Example Usage**:

import qcodes.driver.MDAC as MDAC  
mdac = MDAC.MDAC(name="qcodes\_MDAC", address="ASRL5::INSTR")  
mdac.ch01.voltage(2)

dac\_01\_volts = mdac.ch01.voltage()

**Parameter**: voltage\_raw – R/W

Channel {x} voltage\_raw

**Units**: raw DAC values

**Range**: 0x00000 to 0xFFFFF

**Description**: current output voltage [raw values], set immediately

**Example Usage**:

mdac.ch01.voltage\_raw(0x7FFFF)

ch01\_dac\_value = mdac.ch01.voltage\_raw()

**Parameter**: amplitude – R/W

Channel {x} amplitude

**Units**: Volts peak-peak

**Range**: 0 to 10

**Description**: current waveform's peak-to-peak amplitude (zero for DC)

**Example Usage**:

mdac.ch01.awg\_sine(15,2.5,1) # Set a 15Hz, 2.5Vpp, 1V DC offset sine wave

ch01\_waveform\_amplitude **=** mdac.ch01.amplitude()

**Parameter**: offset – R/W

Channel {x} offset

**Units**: Volts

**Range**: -5 to 5

**Description**: current waveform's offset

**Example Usage**:

mdac.ch01.awg\_sine(15,2.5) # Set a 15Hz, 2.5Vpp, 0V DC offset sine wave

mdac.ch01.offset(1.5) # 1.5V offset

ch01\_waveform\_offset **=** mdac.ch01.offset()

**Parameter**: frequency – R

Channel {x} frequency

**Units**: Hertz

**Description**: current waveform's frequency

**Example Usage**:

mdac.ch01.awg\_sine(15,2.5) # Set a 15Hz, 2.5Vpp sine wave

ch01\_waveform\_frequency **=** mdac.ch01.frequency()

**Parameter**: period – R

Channel {x} period

**Units**: Seconds

**Description**: current waveform's period

**Example Usage**:

mdac.ch01.awg\_sine(15,2.5) # Set a 15Hz, 2.5Vpp sine wave

ch01\_waveform\_period **=** mdac.ch01.period()

**Parameter**: phase – R/W

Channel {x} phase offset

**Units**: Degrees

**Range**: 0 to 360

**Description**: current waveform's phase offset

**Example Usage**:

mdac.ch01.awg\_sine(15,2.5) # Set a 15Hz, 2.5Vpp sine wave

mdac.ch01.phase(90) # Delay the sine wave by ¼ wave

ch01\_waveform\_phase\_offset **=** mdac.ch01.phase()

**Parameter**: waveform – R

Channel {x} waveform

**Units**: Waveform names

**Range**: 'sin', 'tri', 'sqr', 'saw', 'was', 'off'

**Description**: current waveform's type

**Example Usage**:

mdac.ch01.awg\_sine(15,2.5) # Set a 15Hz, 2.5Vpp sine wave

ch01\_waveform\_name **=** mdac.ch01.waveform()

**Parameter**: default\_ramprate – R/W

Channel {x} default ramp-rate

**Units**: Volts/Second

**Range**: 0 to 100000

**Description**: default ramp-rate used for the ramp command

**Example Usage**:

mdac.ch02.default\_rampate(0.05) # Set ch02’s default ramp rate at 50 mV/Sec

mdac.ch02.ramp(2) # ramp from the current voltage to 2V at the default ramp-rate

ch02\_defaultramp\_rate **=** mdac.ch02.default\_ramprate()

**Parameter**: ramp\_rate – R

**Units**: Volts/Second

**Description**: Returns the current ramp rate or None if there is no active ramp

**Example Usage**:

mdac.ch02.ramp(5,0.25) # ramp from the current voltage to 2V at 250mV/Sec

ch02\_current\_ramp\_rate **=** mdac.ch02.ramp\_rate()

**Parameter**: ramp\_destination – R

Channel {x} ramp end-point

**Units**: Volts

**Description**: returns the current ramp destination or None if there is no active ramp

**Example Usage**:

mdac.ch02.ramp(5,0.25) # ramp from the current voltage to 2V at 250mV/Sec

ch02\_current\_ramp\_destination **=** mdac.ch02.ramp\_destination ()

**Function**: ramp(self, destn\_voltage, ramp\_rate=None)

**Description**: Helper function to set a ramp from the current voltage to a specified destination. If a ramp-rate is not specified, the default is used.

Args: destn\_voltage = ramp destination voltage (Volts)

ramp\_rate = ramp rate (Volts/Sec) - (default = default\_ramp\_rate)

**Example Usage**:

mdac.ch02.ramp(5,0.25) # ramp from the current voltage to 2V at 250mV/Sec

**Parameter**: autosync – R/W

Channel {x} waveform auto resync

**Units**: Boolean

**Range**: True, False

**Description**:

True, issues a resync command after a waveform update

False, a resync must be issued using MDAC.sync()

**Example Usage**:

# Set ch05 to issue an mdac.sync()whenever a waveform is set/modified  
mdac.ch05.autosync(True)

ch05\_autosync\_state = mdac.ch05.autosync()

**Parameter**: limit\_max – R/W

Channel {x} maximum voltage limit

**Units**: Volts

**Range**: -5 to 5

**Description**: maximum voltage hardware limit

**Notes**: Only present if MDAC was created with hardware\_limits= True (which is the default)

**Example Usage**:

ch05\_original\_max = mdac.ch05.limit\_max()

mdac.ch05.limit\_max(2.5) # limit ch05 to <= 2.5V

**Parameter**: limit\_min – R/W

Channel {x} maximum voltage limit

**Units**: Volts

**Range**: -5 to 5

**Description**: minimum voltage hardware limit

**Notes**: Only present if MDAC was created with hardware\_limits= True (which is the default)

**Example Usage**:

ch05\_original\_min = mdac.ch05.limit\_min()

mdac.ch05.limit\_min(-0.25) # limit ch05 to >= -250mV

**Parameter**: limit\_ rate – R/W

Channel {x} maximum rate limit

**Units**: Volts/Second

**Range**: 0.2 to 203504

**Description**: maximum rate of change, hardware limit

**Notes**: Only present if MDAC was created with hardware\_limits= True (which is the default)

**Example Usage**:

ch05\_original\_limit\_rate = mdac.ch05.limit\_rate()

mdac.ch05.limit\_rate(0.05) # limit ch05 to voltage changes of <= 50mV/Sec

**Function**: attach\_trigger(self)

**Description**: Assigns channel as trigger source

**Notes**: Each DAC is associated with the SLAVE card that manages it - SLAVE 1 manages DACs 1 to 8, SLAVE 2 manages DACs 9 to 16 etc. Each SLAVE card manages 1 trigger – SLAVE 1 generates TRIGGER 1, SLAVE 2 generates TRIGGER 2 etc. The *attach\_trigger()* function assigns the specified DAC as the trigger source for the relevant SLAVE.

**Example Usage**:

mdac.ch05.awg\_sawtooth(5,0.125)

mdac.ch05.attach\_trigger()# assign TRIGGER\_1 to DAC 5

**Function**: channel\_number(self)

**Description**: DAC channel number corresponding to SMC channel numbers

**Example Usage**:

for ch [mdac.ch01, mdac,ch12, mdac.ch14, mdac.ch15]:

chan\_num = ch.channel\_number()

chan\_volts = ch.voltage()

print(‘Chan {}: {} Volts’.format(chan\_num, chan\_volts)

**Function**: block(self)

**Description**: Blocks until channel is not ramping

**Example Usage**:

mdac.ch01.ramp(1) # ramp DAC 1 to 1V

print(‘Waiting for ramp to complete’)

mdac.block()

print(‘Ramp complete’)

**Function**: awg\_sine(self, frequency, amplitude, offset=0, phase=0)

**Description**: Helper function to set the waveform to a sine wave at a given frequency, amplitude and offset

Args: frequency - [Hz]

amplitude - [Vpp]

offset - [V]

phase - [Deg]

**Example Usage**:

# Set a 5Hz, 2.5Vpp sine wave with a 1V DC Offset and delayed by 180 degrees mdac.ch10.awg\_sine(5,2.5,1,180)

**Function**: awg\_sawtooth(self, frequency, amplitude, offset=0, phase=0)

**Description**: Helper function to set the waveform to a rising sawtooth wave at a given frequency, amplitude and offset

Args: frequency - [Hz]

amplitude - [Vpp]

offset - [V]

phase - [Deg]

**Example Usage**:

# Set a 5Hz, 2.5Vpp sawtooth with a 1V DC Offset and delayed by 180 degrees mdac.ch10.awg\_sawtooth(5,2.5,1,180)

**Function**: awg\_sawtooth\_falling(self, frequency, amplitude, offset=0, phase=0)

**Description**: Helper function to set the waveform to a falling sawtooth wave at a given frequency, amplitude and offset

Args: frequency - [Hz]

amplitude - [Vpp]

offset - [V]

phase - [Deg]

**Example Usage**:

# Set a 5Hz, 2.5Vpp reverse sawtooth with a 1V DC Offset and delayed by 180 degrees mdac.ch10.awg\_sawtooth\_falling(5,2.5,1,180)

**Function**: awg\_square(self, frequency, amplitude, offset=0, phase=0)

**Description**: Helper function to set the waveform to a square wave at a given frequency, amplitude and offset

Args: frequency - [Hz]

amplitude - [Vpp]

offset - [V]

phase - [Deg]

**Example Usage**:

# Set a 5Hz, 2.5Vpp square wave with a 1V DC Offset and delayed by 180 degrees mdac.ch10.awg\_square(5,2.5,1,180)

**Function**: awg\_triangle(self, frequency, amplitude, offset=0, phase=0)

**Description**: Helper function to set the waveform to a triangle wave at a given frequency, amplitude and offset

Args: frequency - [Hz]

amplitude - [Vpp]

offset - [V]

phase - [Deg]

**Example Usage**:

# Set a 5Hz, 2.5Vpp triangle wave with a 1V DC Offset and delayed by 180 degrees mdac.ch10.awg\_triangle(5,2.5,1,180)

**Function**: awg\_off(self)

**Description**: Helper function to turn the waveform off

**Example Usage**:

mdac.ch10.awg\_off() # Stop any currently active waveform on chan 10

**Function**: awg\_arbitrary\_wave(self, arb\_data, phase=0)

**Description**: Helper function to set an arbitrary waveform

Args: arb\_data - an array of 0V offsets in Volts

phase - [Deg]

**Example Usage**:

# Build 150 mSec ARB waveform data in volts

wavelength = 0.15

high\_volts = 4

sample\_count = int(wavelength \* mdac.sample\_rate() + 0.5)  
arb\_waveform\_data = []  
for i in range(0, sample\_count):  
 data\_point = high\_volts \* math.sin(2 \* math.pi \* i / sample\_count)  
 arb\_waveform\_data.append(data\_point)  
  
mdac.ch11.awg\_arbitrary\_wave(arb\_waveform\_data)

**Function**: awg\_arbitrary\_wave\_raw(self, arb\_data, phase=0)

**Description**: Helper function to set an arbitrary waveform

Args: arb\_data - an array of raw DAC data

phase - [Deg]

**Example Usage**:

# Build 250 mSec ARB waveform data in raw DAC sample values

wavelength = 0.25

high\_volts = 2.25

mdac.ch12.voltage(high\_volts)  
max\_offset = mdac.ch12.voltage\_raw() - mdac.ch12.\_zero\_volts\_dac  
sample\_count = int(wavelength \* mdac.sample\_rate() + 0.5)  
arb\_waveform\_data = []  
for i in range(0, sample\_count):  
 data\_point = max\_offset \* math.sin(2 \* math.pi \* i / sample\_count) \* math.sin(  
 32 \* math.pi \* i / sample\_count)  
 sample = int(data\_point + 0.5)  
 arb\_waveform\_data.append(sample)  
  
mdac.ch12.awg\_arbitrary\_wave\_raw(arb\_waveform\_data)

**Function**: \_get\_awg\_arbitrary\_wave\_raw(self)

**Description**: Internal helper function to get the data from an arbitrary waveform

Returns: A tuple of length equal to the number of samples in the waveform

**Example Usage**: N/A

**Function**: \_get\_preset(self)

**Description**: Internal helper function to get the active preset

Returns: \_get\_preset()[0] is the waveform

\_get\_preset()[1,2] is the period in seconds, ticks

\_get\_preset()[3,4] is the offset/ramp\_start in volts, dac units

\_get\_preset()[5,6] is the amplitude/ramp\_destn in volts, dac units

\_get\_preset()[7] is the number of shots

\_get\_preset()[8,9] is the phase offset in seconds, ticks

**Example Usage**: N/A

**Function**: \_set\_preset(self, preset)

**Description**: Internal helper function to set a preset

Args: preset is a tuple consisting of:

- preset[0] is the waveform

- preset[1,2] is the period in seconds, ticks

- preset[3,4] is the offset/ramp\_start in volts, dac units

- preset[5,6] is the amplitude/ramp\_destination in volts, dac units

- preset[7] is the number of shots

- preset[8,9] is the phase offset in seconds, ticks

**Example Usage**: N/A

**Function**: \_get\_dac\_config(self)

**Description**: Internal helper function to get the current dac configuration

Returns: \_get\_dac\_config[0] is DAC update rate in Hz

\_get\_dac\_config[1,2] is max length (samples,secs) for all waveforms

\_get\_dac\_config[3,4] is max length (samples,secs) for RAMPs

The remaining fields are \_reserved\_

**Example Usage**: N/A

**Function**: \_get\_calibration(self)

**Description**: Internal helper function that returns a dictionary representing the calibration data for the channel

**Example Usage**: N/A

**Function**: \_set\_calibration(self, cal\_dict)

**Description**: Internal helper function that accepts a dictionary in the same form as the one given by calling \_get\_calibration(), to alter the calibration data of the DAC

**Example Usage**: N/A

**Function**: \_store\_calibration(self)

**Description**: Internal helper function that stores the calibration to NV memory

NOTE: stores the entire card’s channel calibration data to memory

**Example Usage**: N/A

**Function**: \_store\_limits(self)

**Description**: Internal helper function that stores the HW limits to NV memory

NOTE: stores the entire card’s channel limits to memory

**Example Usage**: N/A

**class MDACChannel(MDACBasicChannel):**

**Description**: Channel Class for MDAC

**Notes**:

Base Channel Class for MDAC DACs.

One instance is automatically created for each DAC in the MDAC chassis

Ie: mdac.chan01, mdac.chan02 .. mdac.chan64

**Example Usage**: N/A - When an MDAC object is created, MDACChannel objects are automatically created for each of the externally available DACs.

**Function**: \_\_init\_\_(self, parent, name, channum)

**Description**: Object creation

Args: parent (Instrument): The instrument the channel is a part of.

name (str): the name of the channel

channum (int): MDAC channel number ([1-64] for populated MDAC)

**Example Usage**: N/A

**Parameter**: divider – R/W

Channel {x} divider

**Units**: On/Off

**Range**: ‘on’, ‘off’

**Description**: dac output voltage divider

NOTE: currently voltage ranges not updated to new range

**Example Usage**:

mdac.ch10.divider(‘on’)

divider\_state = mdac.ch10.divider()

**Parameter**: filter – R/W

Channel {x} filter

**Units**: Filter numbers

**Range**: 1, 2

**Description**:

dac output filter

1 -> (~ 1000 KHz LPF)

2 -> (~ 10 Hz LPF)

**Notes**: The specific filter values are dependent on the hardware installed during manufacture.

**Example Usage**:

mdac.ch10.filter(2)

current\_filter = mdac.ch10.filter()

**Parameter**: microd – R/W

Channel {x} Micro-D relay

**Units**: Open/Close

**Range**: ‘open’, ‘close’

**Description**: micro-D output relay

**Example Usage**:

mdac.ch10.microd(‘close’)

current\_microd\_state = mdac.ch10.microd()

**Parameter**: dac\_output – R/W

Channel {x} output relay

**Units**: Open/Close

**Range**: ‘open’, ‘close’

**Description**: dac output relay onto signal routing common point

**Example Usage**:

mdac.ch10.dac\_output(‘open’)

current\_dac\_output\_state = mdac.ch10.dac\_output()

**Parameter**: smc – R/W

Channel {x} SMC relay

**Units**: Open/Close

**Range**: ‘open’, ‘close’

**Description**: SMC output relay

**Example Usage**:

mdac.ch10.smc(‘close)

current\_cmd\_state = mdac.ch10.smc()

**Parameter**: bus – R/W

Channel {x} BUS relay

**Units**: Open/Close

**Range**: ‘open’, ‘close’

**Description**: signal bus relay

**Notes**: Only one DAC at a time is permitted to close the BUS relay.

**Example Usage**:

mdac.ch10.bus(‘close’)

current\_bus\_state = mdac.ch10.bus()

**Parameter**: gnd – R/W

Channel {x} GND relay

**Units**: Open/Close

**Range**: ‘open’, ‘close’

**Description**: internal grounding relay

**Example Usage**:

mdac.ch10.gnd(‘close’)

current\_gnd\_state = mdac.ch10.gnd()

**class Trigger(InstrumentChannel):**

**Description**: Dedicated trigger channel class for MDAC

**Notes**: Trigger Class for MDAC DACs.

One instance is created for the MDAC chassis

Ie: mdac.trig0

**Example Usage**: N/A - When an MDAC object is created, one Trigger object is automatically created for TRIGGER 0.

**Function**: \_\_init\_\_(self, parent, name, trignum):

**Description**: Object creation

Args: parent (Instrument): The instrument the channel is a part of.

name (str): the name of the trigger

trignum (int): MDAC trigger number ([0] for v1.5 MDAC firmware)

**Example Usage**: N/A

**Parameter**: autosync – R/W

Trigger {x} waveform auto resync

**Units**: Bool

**Range**: True, False

**Description**: True, issues a resync command after a trigger update

False, a resync must be issued using MDAC.sync()

**Example Usage**:

mdac.trig0.autosync(True)

trig0\_autosync\_state = mdac.trig0.autosync()

**Parameter**: frequency – R

**Units**: Hertz

**Description**: trigger frequency

**Example Usage**:

mdac.trig0.frequency(5) # Set TRIGGER 0 to 5Hz

trig0\_freq = mdac.trig0.frequency()

**Parameter**: period – R

**Units**: Seconds

**Description**: trigger period

**Example Usage**:

trig0\_period = mdac.trig0.period()

**Parameter**: phase – R

**Units**: Degrees

**Description**: trigger phase offset

**Example Usage**:

trig0\_phase = mdac.trig0.phase()

**Parameter**: direction: R/W

**Units**: Up, Down

**Range**: ‘up’, ‘down’

**Description**: trigger leading edge direction

**Example Usage**:

mdac.trig0.direction(‘down’)

trig0\_dirn = mdac.trig0.direction()

**Function**: trigger\_number(self)

**Description**: TRIG number corresponding to SMC trigger numbers

**Notes**: For firmware v1.5, this is always 0 – ie: TRIGGER 0

**Example Usage**: N/A

**Function**: start(self, frequency, period=None, phase=None, period\_raw=None)

**Description**: Enables the trigger generation.

Args: frequency - [Hz]

period - [Sec]

phase - [Deg]

period\_raw (available when period is None) - [Sample]

**Example Usage**:

# Configure a 2Hz downward going pulse on TRIGGER 0

mdac.trig0.direction(‘down’)

mdac.trig0.start(2)

# Configure a 2 second upward going pulse on TRIGGER 0

mdac.trig0.direction(‘up’)

mdac.trig0.start(None,period=2)

**Function**: stop(self)

**Description**: Disables trigger generation

**Example Usage**:

mdac.trig0.stop() # Stop TRIGGER 0 from being generated

**class MDAC(VisaInstrument):**

**Description**: Driver for MDAC V1.x - a low noise precision voltage source with integrated signal routing and waveform functionality

HW NOTES: don't exceed 500mA of current through any relay

only toggle relays at zero voltage

don't source or sink more that 20mA of current from any output

don't connect two or more DAC outputs together

in QT configuration, micro-D pin 13 is not connected but pin 25 is.

DAC channel / SMC label 13 maps to micro-D #1 pin 25

similarly, DAC channel / SMC label 45 maps to micro-D #2 pin 25

**Example Usage**:

import qcodes.driver.MDAC as MDAC  
mdac = MDAC.MDAC(name="qcodes\_MDAC", address="ASRL5::INSTR")

**Function**: \_\_init\_\_(self, name, address, baudrate=460800, debug=False,

logging=None, hardware\_limits=True)

**Description**: Object creation

Instantiates the MDAC.

Args: name (str): the instrument name for qcodes

address (str): the VISA resource name

**Example Usage**: N/A

**Parameter**: sample\_rate – R

**Units**: Samples/Second

**Description**: DAC sample update rate for all MDAC channels

**Example Usage**:

DAC\_update\_rate = mdac.sample\_rate()

**Parameter**: bus – R/W

**Units**: Open/Close

**Range**: ‘open’, ‘close’

**Description**: BUS SMC RELAY\n located under the micro-D outputs

**Example Usage**:

mdac.bus(‘close’)

mdac.ch02.bus(‘close’)

print(‘main BUS output is {}’.format(mdac.bus()))

**Parameter**: temperature – R

**Units**: °C

**Description**: system average temperature - a number in the 50s is not unusual

**Example Usage**:

print(‘chassis temperature: {}’.format(mdac.temperature()))

**Parameter**: protection – R/W

Output limits protection

**Units**: On, Off

**Range**: ‘on’, ‘off’

**Description**: globally enables / disables HW based limit protection

**Notes**: Only available if MDAC was created with hardware\_limits= True (which is the default)

**Example Usage**:

mdac.protection(‘off’)

if mdac.protection() == ‘off’:

print(‘WARNING: MDAC over/under voltage protection is DISABLED’)

**Parameter**: supply\_voltages' – R

**Units**: Volts

**Description**: Average of system voltages

(5V, 3.3V, VDD, VSS)

**Example Usage**:

supply\_volts = mdac.supply\_voltages()

print(‘MDAC Supply voltages: 5V = {}, 3.3V = {}, VDD = {}, VSS = {}’ \

.format(supply\_volts[0], supply\_volts[1], supply\_volts[2], supply\_volts[3])

**Parameter**: arb\_engine – R/W

MDAC AWG engine status

**Units**: On, Off

**Range**: ‘on’, ‘off’

**Description**: arb\_engine responsible for updating dac outputs - if off, all waveforms stop and ramps fail to update

**Example Usage**:

mdac.arb\_engine(‘off’)

mdac.ch10.awg\_square(5,2.5)

mdac.ch11.awg\_square(10,2.5)

mdac.ch12.awg\_square(15,2.5)

mdac.arb\_engine(‘on’)

**Function**: sync()

**Description**: Synchronises waveforms running on multiple channels

**Example Usage**:

mdac.ch10.awg\_square(5,2.5)

mdac.ch11.awg\_square(10,2.5)

mdac.ch12.awg\_square(15,2.5)

mdac.sync()

**Function**: run(self)

**Description**: enables arb\_engine - waveforms will update and ramps will continue

**Notes**: run() and arb\_engine(‘on’) are functionally equivalent.

**Example Usage**:

mdac.stop()

mdac.ch10.awg\_square(5,2.5)

mdac.ch11.awg\_square(10,2.5)

mdac.ch12.awg\_square(15,2.5)

mdac.run()

**Function**: stop(self)

**Description**: disables arb\_engine -

all waveforms / ramps will hold at their current value

**Notes**: stop() and arb\_engine(‘off’) are functionally equivalent.

**Example Usage**:

mdac.stop()

mdac.ch10.awg\_square(5,2.5)

mdac.ch11.awg\_square(10,2.5)

mdac.ch12.awg\_square(15,2.5)

mdac.run()

**Function**: list\_triggers(self)

**Description**: lists the dac channels currently generating trigger outputs

**Example Usage**:

slave\_3\_trigger = mdac.list\_triggers()[2]

if (slave\_3\_trigger == -1) and (slave\_3\_trigger != 23):

mdac.ch23.attach\_trigger

**Function**: write(self, cmd)

**Description**: write a command to the MDAC.

checks output and puts it in a queue to be read later

Write has to read a response because of the way the qcodes parameters are implemented: here setting a value gets set using the `write` command. Some instruments like the MDAC always return a status that has to be read.

**Notes**: Writes raw commands to the MDAC – not intended for general usage.

**Example Usage**: N/A

**Function**: query(self, cmd)

**Description**: write a command to the MDAC and return the response

**Notes**: Writes raw commands to the MDAC and returns the raw response – not intended for general usage.

**Example Usage**: N/A

**Function**: wait\_ramps(self, channels=None)

**Description**: Blocks until all channels in the list have finished ramping

'None' will wait for all channels (slow)

**Example Usage**:

mdac.stop()

mdac.ch01.ramp(1) # ramp DAC 1 to 1V

mdac.ch02.ramp(2) # ramp DAC 2 to 2V

mdac.ch03.ramp(3) # ramp DAC 3 to 3V

mdac.run()

print(‘Waiting for ramps to complete’)

mdac.block([mdac.ch01, mdac.ch02, mdac.ch03])

print(‘All ramps complete’)

**Function**: get\_idn(self)

**Description**: generates the \*IDN dictionary for qcodes

**Example Usage**: N/A

**Function**: \_get\_all\_dacs\_config(self)

**Description**: Helper function to get the active preset

Returns a list of up to 65 items for dac\_0 .. dac\_64

Each channel's list entry consists of:

[0,1] the current output in volts, dac units.

[2] whether the current channel has been limited.

[3] a list describing the current waveform:

- wave[0] is the name. If 'off', the list stops here.

- wave[1,2] is the period in seconds, ticks.

- wave[3,4] is the offset/ramp\_start in volts, dac units.

- wave[5,6] is the amplitude/ramp\_destination in volts, dac units.

- wave[7] is the number of shots.

- wave[8,9] is the phase offset in seconds, ticks.

[4] describes the current relays (None for dac 0):

- if not None, [main, smc, bus, term, f1, bypass, dac\_output, divider]

**Notes**: This function is intended for use during a snapshot, not necessarily during general operation.

**Example Usage**: N/A