Author		Date	Document no
JKU/AK		27-04-2021	PM10091-14
Approved	Date	File / reference	
AK	27-04-2021	PM10091-14 QDAC sw1.07.docx	

QDAC 1.0

User manual - 24 channel* precision voltage source

WARNING

This instrument is a sensitive laboratory apparatus. Please read the manual, in particular the command descriptions, and always follow the checklist before turning on power. The instrument may be damaged if the start-up checklist is not followed.

Scan QR code to access drivers and manuals:

*) A 48 channel version is available on request and has been delivered to a few customers.

Author	Date	Document no
AK	27-04-2021	PM10091-14

Table of contents

1 Introduction

The QDAC series from QDevil are high-precision low-noise computer-controlled voltage generators designed for controlling gates in quantum electronic experiments. The standard QDAC has 24 output channels, but a version with 48 channels is manufactured on request and has been delivered to a few customers. This manual covers both instruments. if you have the 48-channel version, then whenever the number 24 is mentioned in this manual, think “48”. This is also true for the software commands.

1.1 Overview

The output channels are labelled “CH 01” to “CH 24”. Each channel can either output a DC voltage or a waveform. In both cases the bit resolution and maximum voltage are defined by the voltage range (nominally ±10V or ±1.1V) which can be configured individually for each channel. During normal operation all channels are updated synchronously once every millisecond.

The outputs are designed for voltage control of high-impedance devices but can nominally supply up to 1mA in the 10V range and 10μA in the 1.1V range. The special high current model of the QDAC (Q303) can source currents up to ±10mA within specs. Please see Appendix F.

Each channel has an A/D converter for current measurement, which allows sensing of small currents. This is useful for detecting unintentional shorts or leakage currents of the device under study, for example between bonding wires or gate electrodes.

The unit contains ten waveform generators: Eight predefined waveform generators (sine/square/triangle/staircase with configurable parameters), one pulse-train generator, and one user-defined arbitrary waveform generator (AWG). The user-defined waveform can hold up to 8000 points, corresponding to 8 seconds. The waveforms of the function generator can be output on one or more output channels with a voltage amplitude and offset selected individually for each channel.

For synchronizing external equipment with any of the waveform generators there are several configurable SYNC outputs, which will output a pulse at a configurable delay after the (re)start of a selected waveform generator. To synchronize software control, the QDAC can also send a synchronization message on the communications channel to the connected computer.

As an option a sync-board extension is becoming available. This will allow multiple QDACs to work synchronously and enhance the possibilities of synchronizing clock and generators with external equipment.

Each row of BNC connectors belongs to one printed circuit board. There are three temperature sensors on each board. They can be read out by software commands. This can in principle be used to compensate laboratory temperature fluctuations, by reading out the temperature and changing the voltage accordingly.

The QDAC is controlled through a USB/serial interface using simple commands easily used from any programming language. QDevil supplies a Python library and examples for straightforward integration with Python programs. Drivers for QCoDeS and Labber instrument control softwares are also available.

1.2 The front panel

CH01 – CH24

Each BNC connector output a programmed voltage in ranges ±10V or ±1.1V with a 20bit precision (19.0735µV resolution in 10V mode) limited only by the precision of the 1ppm internal reference and the tolerances of the carefully selected components in the output circuit. Each of the voltage outputs has a sensitive current measurement A/D converter, so that it is possible to detect and quantify for example leaking gates, se below.

SYNC

In the right-most column of BNC connectors there is a number of independently programmable and assignable synchronization (SYNC) outputs ”SYNC”. These are used for synchronization with other instruments and are galvanically isolated from the output channels. For the standard 24-channel unit there are two SYNC outputs. For the 48-channel unit there are five SYNC outputs.

CAL

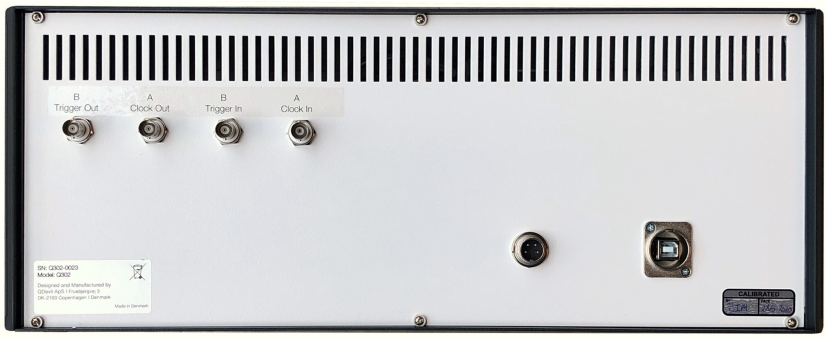
The bottom BNC connector in the right column with the label “SYNC # / CAL” is by default a calibration output. Any of the 1..24 channels can be routed internally to this output via software-controlled relays. Hence, the CAL output is ideal for automatic testing of the performance and function of each channel, with just a single multi-meter attached to the instrument. The reason why the label says “SYNC3 / CAL” (or “SYNC5 / CAL”) is that it is possible to convert this output to a SYNC channel with a small hardware modification (jumper setting inside the QDAC). Contact QDevil if you should be interested in such a modification.

LED

The small LED in the front panel flashes red-green during start-up for about 10 seconds. During normal activity it flashes green. At external command activity it flashes red.

1.3 Rear panel

The rear panel provides air slots for ventilation (for noise and reliability reasons, there are no active ventilators in the QDAC), a connector for the external power supply (Amphenol 4 way) and the USB communication port, and 4 BNC connectors for inter-QDAC synchronization or use of external clock. On some intermediate models these SYNC BNC holes are blinded, indicating that the QDAC can be upgraded with the Sync Board option.



1.4 Voltage and current output ranges and limits

Each output can be in either a ±10V range or a ±1.1V range, nominally, and can be either in DC mode or in a waveform mode where it is outputting a curve from a generator, see below. The voltage range is set using the ‘vol’ usb/serial command. It is recommended to always set the output to zero before changing the voltage range as unexpected steps or spikes may otherwise occur (see ‘vol’ command description).

Note that the applicable voltages may be a bit off the nominal ranges of ±10V and ±1.1V due to the tolerances of components. The actual limits are ± (10±0.002) V and ± (1.1111±0.0004) V, respectively.

The DC voltage is set using the ‘set’ usb/serial command. The output mode is controlled by the ‘wav’ command. At power on the QDAC will be in the ±10V output mode.

The QDAC is designed to supply up to ±1mA (10V range) or ±10μA (1.1V range) at maximum output voltage in appropriate loads.

However, in loads with a small resistance the QDAC can actually deliver a little more than 10mA. It is, however, not recommended to do that, as this will generate significant heating in the output circuits leading to enhanced drift. Therefore, one should also not for example short-circuit the outputs while a voltage is present.

The special high current version of the QDAC (Q303) can supply up to ±10mA within specifications, see Appendix F. The current sensing resolution is smaller for this model.

> Please avoid short circuiting the outputs for a prolonged period in order to avoid stress components unnecessarily and to avoid causing enhanced drift.

1.5 Current measurement

The built-in current sensing feature can detect very small changes in current for each channel. It has an integration time of about 0.2 seconds and a 0.15 second dead time and features 50 Hz and 60 Hz power

line cycle suppression. There are two measurement ranges, which are nominally ±1μA and ±100μA, which are the ranges referred to in the command set[1]. However, the practical ranges are only about ±0.75μA and ±75μA, respectively. In the ±1μA range, current steps down to 0.2 nA, and possibly even smaller, can be resolved. In the ±100 μA range steps down to just under 5 nA can be resolved. Current is read using the 'get" usb/serial command.

> Note that the current sensor has an input filter RC filter which limits its response time. Therefore, one should wait 100-200 ms before taking a measurement after changing the voltage of that channel.

The 100μA range has a limited accuracy for large impedances and may be up to 25% off (plus an offset of 10μA) at impedances > 10 MΩ, whereas the 1μA range is accurate to better than 5% of the reading plus an offset of up to about 0.01μA.

Note that the 100μA range is only available for the 10V output range due to balancing of the power consumption and heat generation in the internal relays in the QDAC. The current range is set using the 'cur' usb/serial command.

	1 μA	100 μA
1.1V	X	
10 V	X	**X**

Allowable combinations of voltage and current ranges. Note that 1 μA and 100 μA are the nominal current ranges, which in practice are limited to about 0.75 μA and 75 μA, respectively[1].

1.6 Waveform generators

The QDAC offers 10 waveform generators:

- 8 predefined waveform generators (sine/square/triangle/staircase)
- 1 arbitrary waveform generator (awg)
- 1 pulse train generator

Each generator can be output on as many channels as wanted with an individual offset voltage and amplitude for each channel. It is the configuration of each individual channel which determines if it is hooked up against one of the generators with its own individual offset and amplitude, or if it just outputs a DC voltage. This is controlled by the 'wav' usb/serial command. If the offset and amplitude defined by the 'wav' command result in a channel output exceeding the voltage range of the channel, the QDAC will respond with an error or in any case clip the signal.

Common to all generators is that they can be programmed to repeat a given number of times, or indefinitely. Each generator is either started the moment it is configured or by a trigger command.

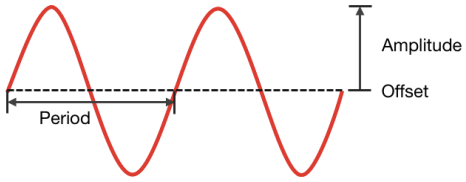
[1] Note that the special high current model of the QDAC (Q303) has a modified output circuit allowing it to source currents up to ±10mA within specs. The current sensing range is increased to ±7.5mA. Please see Appendix F.

1.6.1 Pre-defined waveform generators (1-8)

Each of the eight pre-defined generators, numbered 1-8, can be programmed to output either a sine wave, a square wave, a triangle, or a staircase. This is done using the 'fun' usb/serial command. The amplitude and offset given by the 'wav' command are absolute volts. Their exact meanings can be seen in the drawings on the following pages.

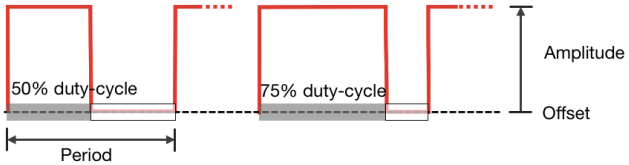
Sine

The sine waveform is for example useful for applying modulation to gate electrodes in a quantum device in order to measure the coupling between gates or between quantum dots using digital lock-in detection. The sine waveform is when suing the 'fun' command defined by its period only. The amplitude given by the 'wav' commands corresponds to half the peak-to-peak swing, and the offset to the mean value.



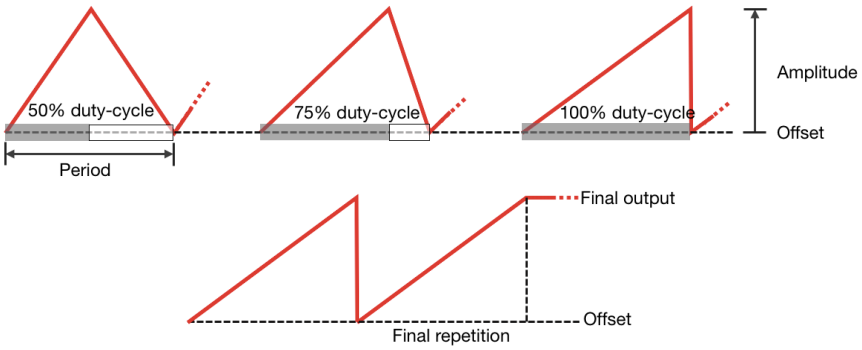
Square wave

Square waves are useful for repeatedly switching a device between two configurations by switching some electrodes on and off but can also be useful as a substitute for digital markers, for synchronization purposes, or for various diagnostic applications. The square wave form is defined by its period and by its duty-cycle. The offset provided in the 'wav' command defines the bottom level of the square wave, and the amplitude defines the height.



Triangle/ramp

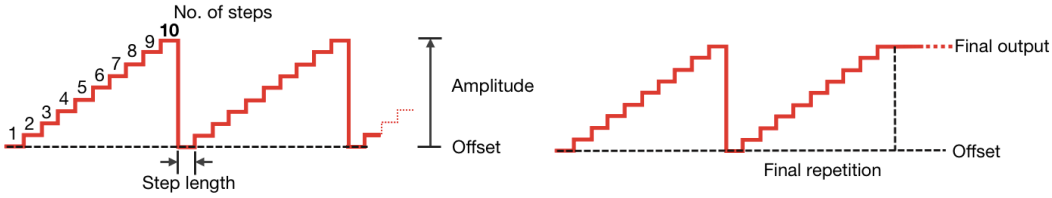
The triangle generator is intended for ramping. It is defined by its period and by its duty-cycle. With a 50% duty cycle the sweep up and sweep down are of equal length. A duty-cycle of 100% will give a rising saw-tooth (0% will give declining saw-tooth). As the QDAC sampling interval is 1 ms, triangles will actually, when zooming in, be staircases with step lengths of 1 ms. The offset given in the 'wav' command defines the starting level of the triangle, and the amplitude defines the height relative to the offset.



Note, that if the triangle waveform is used for ramping (100% duty-cycle), then after the specified number of repetitions the output will stay at the highest value of the ramp. If the generator is stopped by the user the output will jump to the offset value.

Staircase

The staircase waveform is basically a triangle with a duty-cycle of 100%, but with a defined number of steps of well-defined lengths. Hence, the period and lengths of all steps are unambiguously defined. The shortest step length is 1ms corresponding to the sampling rate of the QDAC. Similar to the triangle waveform the offset given in the 'wav' command defines the voltage level of the first step, and the amplitude defines the level of the highest step relative to the offset.



The staircase is intended for sweeping like the triangle waveform but in a discrete way. This can for example be useful for two-dimensional scanning where a staircase in the fast sweeping direction allows for a little bit of integration time at each point. Using a stair case, the slow sweeping channel will hold a constant voltage for each sweep of the fast channel.

After the specified number of repetitions, the output will stay at the final value of the staircase. If the generator is stopped by the user, the output will jump to the offset value.

1.6.2 Arbitrary Waveform Generator (AWG)

Generator no. 9 is an arbitrary waveform generator (AWG). It can hold up to 8000 user-defined samples, each of one 1 ms duration. The samples (in Volts) are uploaded to the QDAC using the 'awg' usb/serial command. The 'run' command is used for starting the generator and for defining triggers and repetitions.

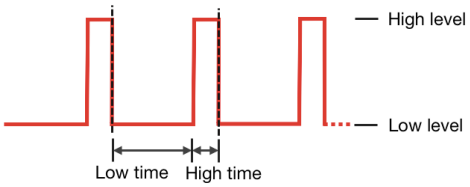
The resulting output voltages are controlled by the 'wav' command. Unlike for generators 1-8 the 'Amplitude" is here a scaling factor not an absolute voltage:

Channel output = Amplitude [] * AWGSignal [V] + Offset [V]

The AWG is intended for cases where the predefined waveforms are not flexible enough.

1.6.3 Pulse Train Generator (PTG)

The pulse train generator is a more accurate version of the square wave pre-defined waveform where it is possible to set directly the length of the low time and the high time parts of the curve and the voltages of the two levels. So, the purpose is the same. It can also be used as an alternative to the SYNC outputs, though it appears on the non-isolated channel outputs.

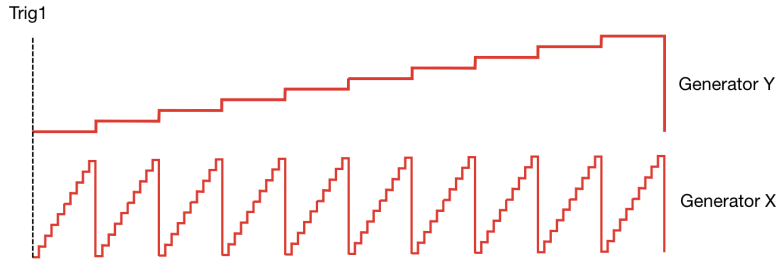


As for the AWG, when a channel is connected to this generator using the ‘wav’ command, ‘Amplitude” is a scaling factor not an absolute voltage.

Channel output = Amplitude [] * PTGSignal [V] + Offset [V]

1.6.4 Synchronised sweeping using triggers

By using triggers several generators can be started simultaneously. This is for example useful for two-dimensional sweeping, e.g. when recording charge stability diagrams in quantum dot arrays: A slow ramp or staircase can be produced by one generator while another generator is producing fast ramps or staircases while an external digitizer is recording the sample response.



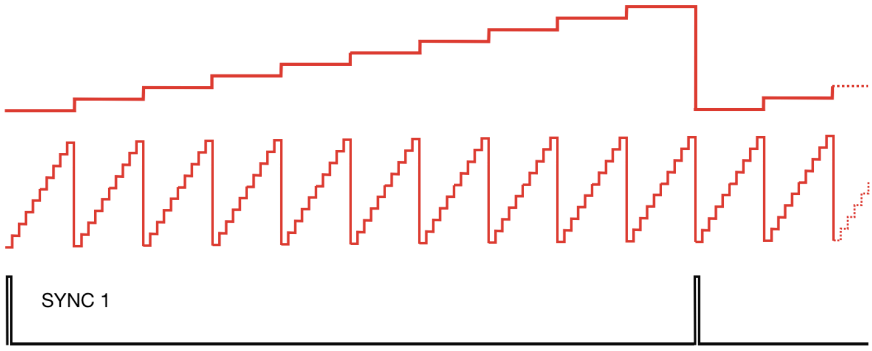
The QDAC has 10 triggers numbered 1-10. When a generator is configured using either the ‘fun’, ‘run’, or ‘pul’ command, it will usually start immediately. However, if a trigger number (>0) is specified, the generator will enter an armed state and will start only when that corresponding trigger is *fired*. In this way several generators can be started simultaneously. Triggers are fired using the ‘trig’ usb/serial command.

When the optional sync-board is installed a trigger can be instructed to wait firing until a pulse appears on the Sync B input on the sync-board.

1.6.5 Synchronizing external equipment

For synchronizing external equipment with one of the waveform generators there are 2-5 configurable SYNC outputs, which will output a pulse at a configurable delay after the (re)start of a selected waveform generator. These ports are galvanically isolated (with an isolation resistance of 1 MΩ) from the QDAC electronics and ground so that ground loops and noise injection is avoided. The signal level is 5V in a high impedance load. In 50 ohm the voltage will reduce to about 1.1 V, corresponding to an output impedance of about 180 ohm.

A pulse with a specified length will appear on the front panel SYNC BNC-connector after an optional delay whenever the waveform of the generator linked to the SYNC connector is repeated. SYNC ports are configured using the 'syn' usb/serial command.



Example. For recording charge stability diagrams of a quantum dot array one can have two generators outputting ramps or staircases, one fast and one slow, on two channels. A digitizer can be triggered to start capturing by a sync pulse output by the QDAC appearing at the start of the scan – when the slow ramp starts. Alternatively, a sync pulse could be output for each "scan line", i.e. at the beginning of the fast generator ramps. The digitizer should be configured to record samples at a speed commensurable with the sweep rates.

To synchronize software control, the QDAC can also send a synchronization message on the communications channel to the connected computer. This is defined using the 'ssy' command.

Sync-board option

For QDACs with the sync-board option, four BNC connectors are located on the back side of the unit: two inputs and two outputs. The inputs are galvanically isolated from the QDAC electronics and the cabinet, so that ground loops are avoided when inter-connecting multiple QDAC or when connecting to the output of external equipment.

Sync-A Out	3.3V digital output for the sample clock (1kHz) of the device. Typically connected to the Sync-A In on slave QDACs . This output is configured using controlled by the "synA" usb/serial command.
Sync-A In	3.3 V digital sample clock input. On slave QDACs this is connected to the clock output (Syn-A Out) on the master. Used by the "extclk" command.
Sync-B Out	3.3 V digital output for synchronising generator triggers. When enabled any trigger generate pulse on this port. Controlled by the "synB" usb/serial command.
Sync-B In	3.3 V digital generator trigger input. Generator triggers will wait for a pulse on this input when a parameter "1" is appended to the "trig" command.

One pair of input/output is reserved for sample clock synchronization between QDACs (Sync-A) but may even be controlled by a third-party oscillator up to 2kHz. Note, that all timing mentioned in this manual relies on a clock frequency of 1kHz. If an external clock is different from this, all “milliseconds” mentioned in this manual will have to be translated to the new clock rate. Note, that if the QDAC is commanded to use an external clock and no external clock is present, the updating of the outputs will stop.

The other pair (Sync-B) is reserved for generator triggering across QDACs. Note that it only makes sense to synchronise generators across QDACs if the sample clocks are also synchronised.

1.7 Calibration

Each output channel and each current monitor of every unit has been calibrated in the laboratory at QDevil. The calibrations have taken place at environment temperatures around 23 °C after having been powered on for more than 8 hours.

It is possible to re-calibrate and adjust the QDAC. The calibration constants of both the voltage outputs and the current sensors are just an offset and a coefficient. Calibration has to be performed for each range of each channel.

The recommended way to calibrate the voltage output, is to sweep through a range of raw DAC values (use the ‘dac’ command) and measure the corresponding output with a high precision volt meter and derive the offset and coefficient from a linear fit to the data. The offset parameter is the raw DAC value (real numbered) for which the output is zero volts. The coefficient is in units of raw DAC values per Volt. Please see the description of the ‘vcal’ command.

To calibrate the current sensors, connect a known resistor similar to the expected load (for example 250 kΩ) to the output and sweep the output voltage (‘set’ command) while reading the current in raw ADC values (‘adc’ command). The offset parameter is the raw ADC value (real numbered) when the actual current is zero (the voltage over the resistor is zero). The coefficient is the number of micro-amperes per raw ADC value (also real numbered). Please see the description of the ‘ical’ command.

To automate calibration, the “’CAL” output BNC can be used for successively connecting it to each of the 1-24 channels (‘cal’ command).

After adjusting the calibration parameters, the should be persisted in flash memory using the ‘savecalib5599’ command in order to survive power down or restart.

> Note, that there is no command for restoring the factory calibration parameters.

2 Specifications

2.1 Table of specifications

Number of channels	24 (or 48 for the special version)
Voltage ranges	±(10±0.002) V and ±(1.1111±0.0004) V, configurable by software on each channel individually
Voltage resolution	20 bits: 1 LSB = 2.1μV (1.1V range) 1 LSB = 19μV (10V range)
Temperature coefficient	< 3 μV/°C @ 25°C, 0.1V output (10V range)
Stability	< ±2 μV fluctuations in 10 hours (10V range) after being on for minimum 48 hours (temperature stable within ±0.3°C)
Maximum integral non-linearity	1.7 LSB (typical)
Maximum differential non-linearity	0.68 LSB (typical)
Current output, nominal[1]	±1mA (10V range) ±10μA (1.1V range)
Current output, out of specs (in low impedance, not recommended)	> ±10 mA (100 μA current sense mode) > ±100 μA (1 μA current sense mode)
Update rate	1 kHz
Predefined waveform generators	8 (sine/triangle/ramp/square)
Arbitrary waveform generator	Up to 8000 samples / 8 sec
Pulse train generator	1
Sync outputs	2 (standard 24-channel unit), 5 (48-channel unit)
Current sensing (standard model) [2]	±75μA (10V and 1.1V range), accuracy (absolute) ≈ 25% + 10 μA ±0.75μA (1.1V range), accuracy (absolute) ≈ 5% + 0.01 μA
Current resolution (standard model) [2]	24 bits, 200 ms settling time, Minimum detectable current step: ±75μA range: < 5nA ±0.75μA range: < 0.2nA
Power requirements[3]	±15V (24-chan: 0.8A, 48-chan: 1.5A). Min-max range: ± (14.5-15.5) V 5.5V (24-chan: 1.5A, 48-chan: 3A). Min-max range: 5.4-5.7 V.
Dimensions (incl. feet)	24-chan. unit, cm: 45 x 32 x 18.5 (48-chan. unit, cm: 45 x 32 x 27.5)
Weight	24-chan unit: 5.5kg (48-chan: unit: 7.7kg)

[1,2)] In the high current model (Q303), the maximum nominal current is ±10mA, see Appendix F for details incl. current sensing.

[3)] These are the requirements for when operated within specifications. If maximum out-of-specs current is drawn from all outputs (up to 15mA), the draw on the ±15V supply may reach 1.6 A on the 24-chan model, and 3 A on the 48-chan model. The high current model will also draw the higher currents when al channels are fully loaded.

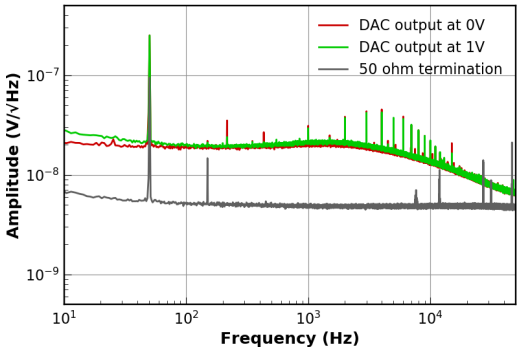
2.2 Output noise and drift

There is a temperature dependence on the voltage outputs, about 3 μV/°C (temperature outside the cabinet). It is also known that the drift is of the order of a few microvolts over 24 hours in a normal laboratory environment after the instrument has been allowed to warm up. Full warm-up may take several days: Even after 48 hours of operation a continued small increase in internal temperature has been measured.

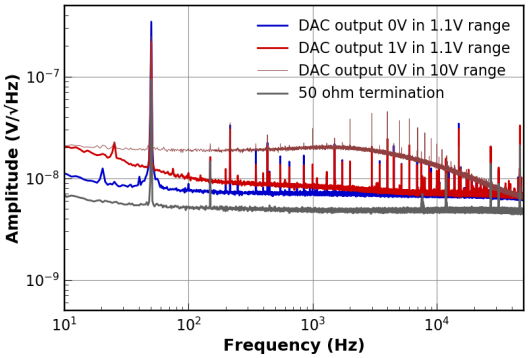
Below, noise spectra are shown for the QDAC when set to an output of 0 V and 1 V in its 10 V range as well as in its 1.1 V range. The data is shown without subtraction of background noise. Except for the 100kHz

low-pass filter in the SR560 pre-amplifier used before the digitizer, the presented data are unfiltered outputs. In in experimental situations external low pass filters are highly recommended.

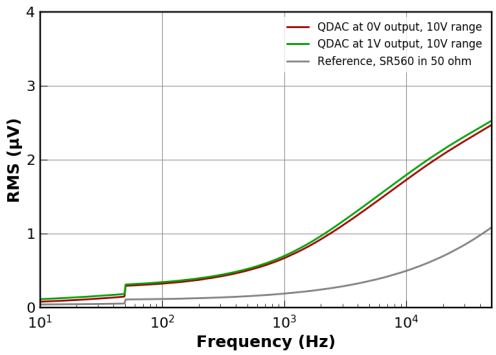
Besides the 50Hz mains frequency and peaks related to the QDAC sampling frequency of 1 kHz, some other peaks are noticeable in the spectra. Many of them seem to come and go and are therefore ascribed to sources in the building, possibly mixed with the sampling frequency of the digitizer or of the QDAC itself. The power in all peaks except for 50 Hz is vanishing.



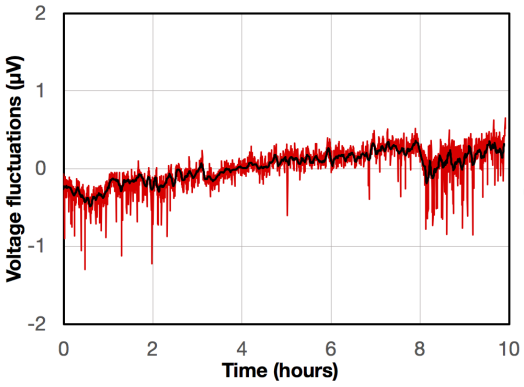
Noise spectra when the output of the QDAC is set to 0V (red curve) and to 1V (green curve) in the 10V range. The grey curve shows the spectrum for a 50 ohm termination.



Noise spectra at 0V (blue) and 1V output (red) in the 1.1V range and for comparison at 0V output in the 10V range (dark red). The grey line is the background noise of the measurement circuit.



Integration of the noise spectra at 0V and 1V output in the 10V range showing the total RMS noise at various frequencies up to 50 kHz. Note that low frequency noise below 10 Hz is not included fully as it has not been recorded.



Voltage fluctuations (DC - 1.5 Hz) over time in the ±10 V range recorded in the QDevil office after 48 hours of on time.

Author	*Date*	*Document no*
AK	27-04-2021	PM10091-14

3 Connecting and powering on the QDAC

It is recommended to install the QDAC in a normal laboratory environment (i.e. at non-extreme tempera–tures and humidity). In order to achieve minimal noise (e.g. 50/60 Hz), the QDAC unit does not have a built-in power supply. Hence, an external power supply is required. In all cases place the power supply as far away from the QDAC as the cable permits. Due to the relatively high current draw and in order to avoid significant voltage drop, the power cable delivered with the instrument is only 3 meters long. Avoid grounding the power supply to racks etc through its cabinet. Only ground it through the mains ground.

3.1 Using the QDevil Power Supply

Please first read the instructions included with the QDevil Power Supply and please make sure that the Input Voltage Range selectors have been set correctly. After inserting the mains power cable to the QDAC and to a grounded mains outlet, make sure that the power switch on the back side of the power supply is in its off position. Then connect the QDAC to the power supply using the enclosed power cable with the 4 pole Amphenol connectors. Finally switch on the power supply. All three green LEDs should light up and the QDAC LED should start flashing red/green.

To shut down the QDAC, simply press the switch on the back side of the power supply to its off position. **Please wait at least 5 seconds after powering off before powering on again.**

QDevil Power Supply. Rear panel enlarged on the right. Please consult the QDevil Power Supply manual for details.

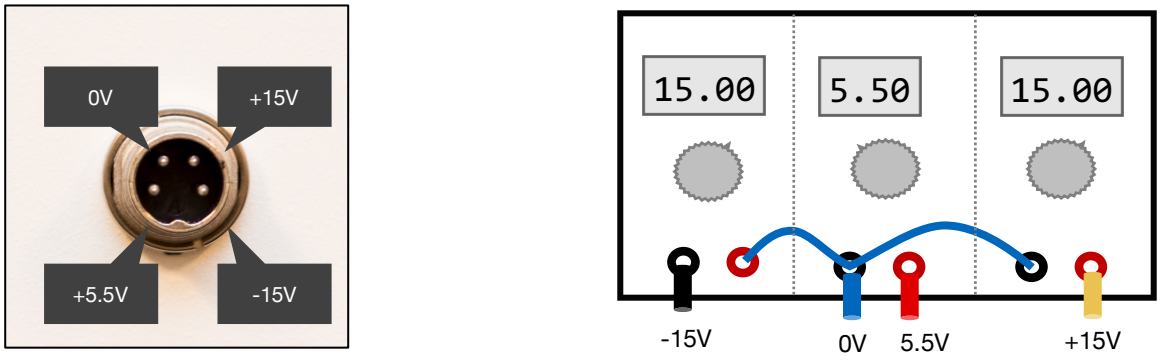
3.2 Using a laboratory power supply.

QDevil recommends using a good laboratory power supply which can deliver nominally ±15 Volts and +5 Volts with a common 0V reference galvanically isolated from mains ground. **For QDACs delivered before August 2019 the power supply *must* have current limiters.** QDevil has good experience with the QL355TP from Thurlby Thandar Instruments. Some customers have good experience with the Keysight E36313A. This is in all cases recommended.

Voltage input (V)	Min (V)	Max (V)
5.5	5.4	5.7
-15	-15.5	-14.5
+15	14.5	15.5

Required input voltages

Use the enclosed combined 15V/5.5V power cable for connecting the QDAC unit to the power supply. Should you want to make your own cable the polarities of the connector are shown in the figure below. Leads with a high cross section area should be used (> 1 mm2). It is recommended to place the external power supply as far away from the QDAC as the power cord allows.



Left: Pinout of the Amphenol power connector on the rear.

Right: Schematic of how the three different voltage output of the power supply should be wired, with the common 0V reference established by the wire drawn in blue (VERY IMPORTANT). The same wiring applies even if it is three separate power supplies.

The standard 24-channel unit will pull about 0.8 A on the ±15V supplies and up to about 1.5A on the +5.5V supply, when operated within specifications. The high current model will pull more current out of the ±15V supplies if all channels are fully loaded, see Appendix F. *Even for the standard model, if all outputs are loaded with small resistance (which is not recommended), the current may go up to 1.6 A ±15V*. The 48-channel special version consumes a little less than twice these amounts.

Please follow the instructions in the procedures in Appendix A and Appendix B.

> If you are **not** using a power supply with a single power-*on* switch make sure to always power 5.5V on first, then +15V, and within 2 seconds -15V.
>
> **Always wait 2 seconds after power *off* before powering *on* again.**

3.3 Connecting the QDAC to your experiment

It is advised not to connect your quantum devices to the QDAC before power-up. You may even want to perform your own initialization. After power-up all output channels will be in the 10 Volt range and will after about 15 seconds be initialized to zero volts using the internal calibration. In the first 15 seconds there may be up to a few milli-Volts on the outputs. As the temperature in your lab may differ from the temperature during calibration, you may observe a small offset (some micro-Volts), even after the first 15 seconds.

> ALWAYS disconnect your experiment from the QDAC before powering off, as the outputs may drift a little bit up and down when power is switched off – not much (NOT to ±10V), but perhaps enough to cause damage to the devices

3.3.1 Filtering

The voltage outputs (BNC 01-24) are internally lowpass filtered with a cut-off frequency of around 5 kHz. This means that the 1kHz update rate is visible in the spectrum, see section 2.2. The reason why a filter with lower cut-off has not been chosen is to allow relatively steep voltage steps.

It is recommended to further low-pass filter the signals before they reach the device under test. For high impedance ultra-low temperature applications, we recommend that this is done using a QFilter from QDevil. The QFilter is designed for low pass filtering and electron thermalization at dilution refrigerator mixing chamber plate temperatures. Alternatively, and depending on your application, you may benefit from placing low-pass filters (e.g. in-line BNC filters) directly on the QDAC outputs.

3.4 Connecting a computer to the USB/serial interface

Connect your computer to the USB connector on the rear panel of the QDAC using the USB cable delivered with the instrument. The QDAC communication works through a virtual serial port over the USB interface, as there is a USB-to-serial adapter in the QDAC unit (opto-coupled). Since the USB connection is opto-coupled, no transients in the output voltages will occur when you plug in or unplug the USB cable. So it is safe to do while a sample is connected to the QDAC.

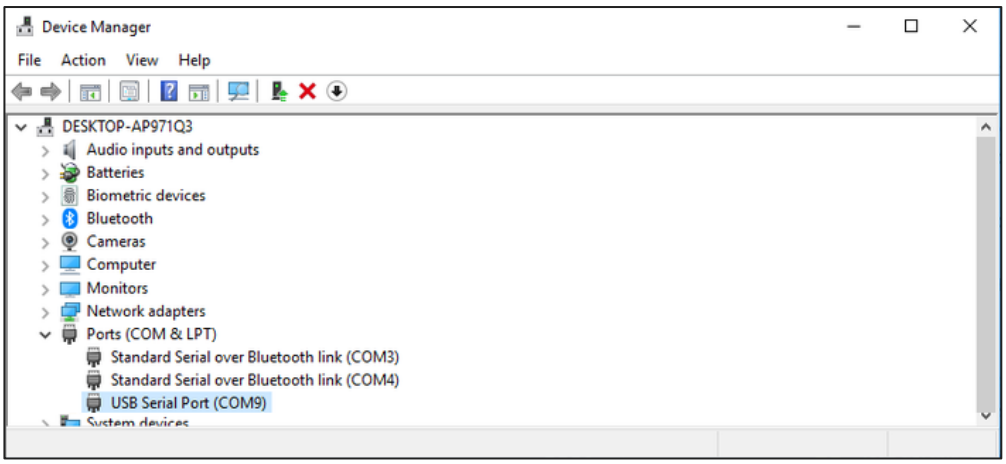
When communicating with the QDAC it is necessary to specify the correct serial port to the software. See how to find the correct port number or name below.

Beware

On Windows computers the port number may in rare cases change when the USB cable is unplugged and plugged in again, or if the computer is rebooted - especially if a large number of serial devices have been connected to the computer.

3.4.1 Windows

On Windows systems the port can be found by going the Device Manager and look under *Ports (COM & LPT)*. In the example below it is seen that USB serial port is found at “COM9”.



The driver is usually pre-installed, or Windows will download the driver when the QDAC usb cable is plugged into the PC. **If this does not happen the VCP driver can be downloaded from ftdi:** https://www.ftdichip.com/Drivers/VCP.htm.

3.4.2 MAC OS and Unix/Linux

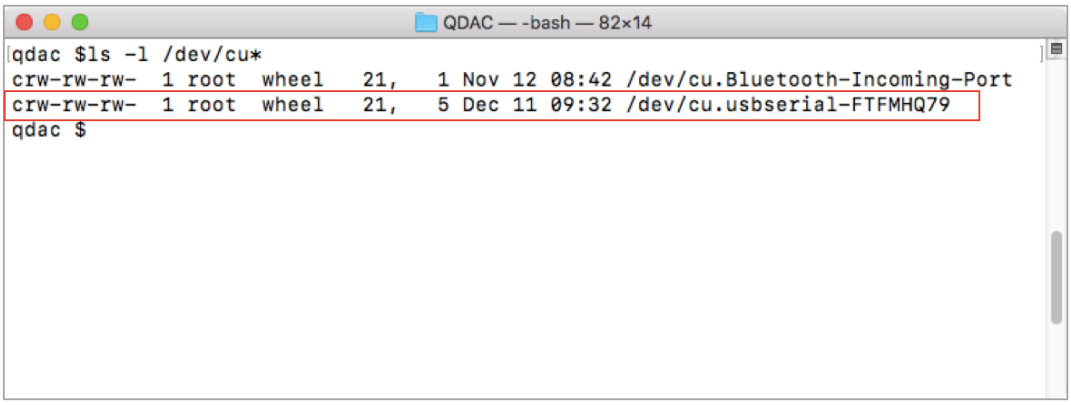
Open a command prompt (terminal) and browse the */dev* folder for all “tty” devices. Use this command to do so:

```
ls -l /dev/cu*
```

or:

```
ls -l /dev/tty*
```

In this example the port ID is “/dev/cu.usbserial-FTFMHQ79”.

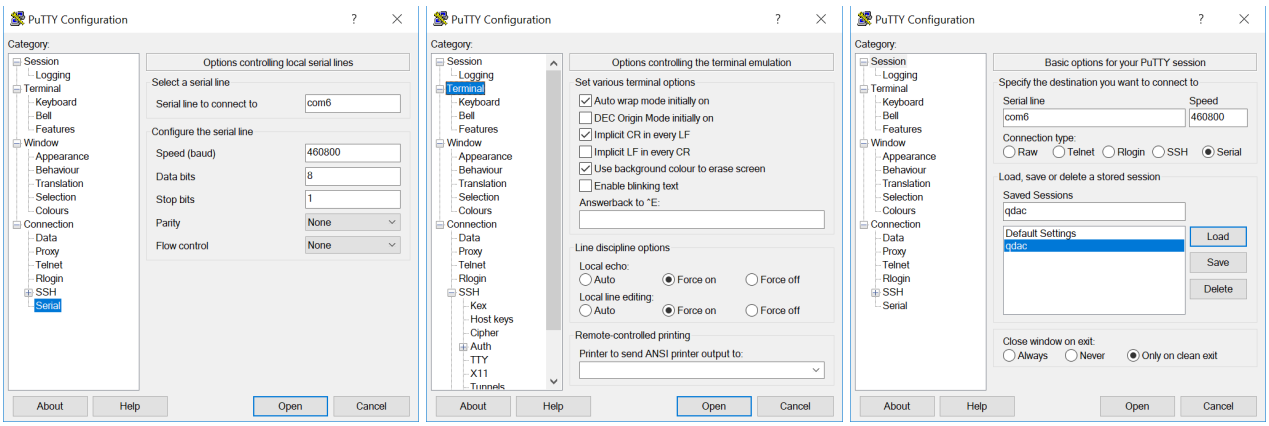


If the usb-serial driver does not appear in the list it may be necessary to install it. The “VCP” driver can be found here: https://www.ftdichip.com/Drivers/VCP.htm.

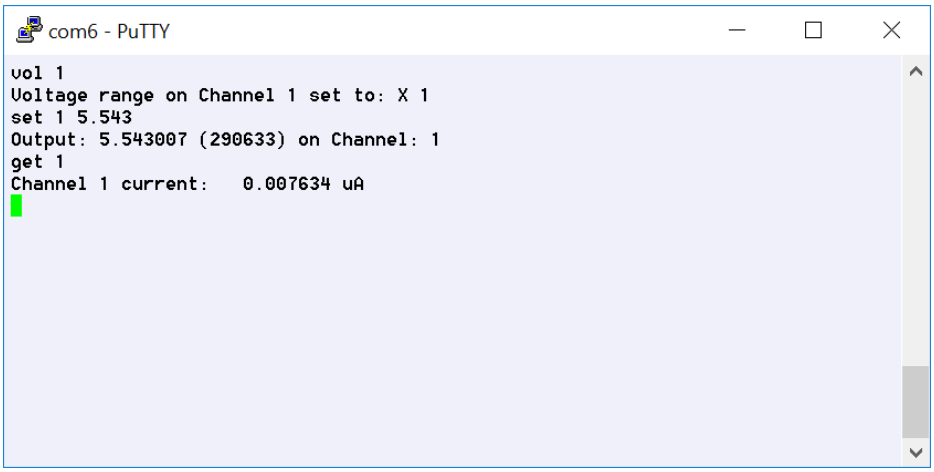
Example

One can communicate with the QDAC through a simple terminal program, like PuTTY or HTerm, using the usb/serial commands described in Appendix C. This is a great way to get to know the QDAC.

Set up the terminal program to use the correct port, select a baud rate of 460800, 8 bits, 1 stop bit, no parity, no flow control, and tell the terminal program to send every time you hit <enter> and to make a newline (CR) for every LF character. It sometimes requires a bit of trial and error to set up these terminal programs….



In this example, using PuTTY) we first ask for the voltage range currently set for CH01 and get a reply. Then we set the voltage output to 5.543 Volts, and finally we read the current output on CH01.



3.4.3 Alphabetical list of USB/serial commands

See appendix Appendix C for detailed descriptions. Commands supported by firmware version 1.06.

Command	Description
adc	Reads the raw, binary, AD converter output from one of the current sensors
awg	Arbitrary Waveform Generator. Sets the user-defined waveform.
brt	Sets the baudrate of the communication channel
cal	Set the channel to be routed to the CAL connector
cur	Sets or reads the range of the current measurement ADC for a selected channel
dac	Sets or reads the raw binary DA converter input
extclk	Use external sample clock (requires sync-board option)
fun	Sets or reads the configuration of one of the eight pre-defined waveform function generators
get	Reads the current measurement ADC or a given channel
ical	Sets or reads current calibration constants
pul	Sets the configuration and starts/arms, or reads the configuration of the pulse train generator
rang	Reads the minimum and maximum calibrated output values of a channel
run	Starts or arms the arbitrary waveform generator
savecalib5599	Stores the calibration parameters in flash memory
set	Sets or reads the DC voltage output of a selected channel
sernum	Reads the serial number of the QDAC
ssy	Software synchronization. Returns at the beginning of the next period for the selected generator.
status	Returns the firmware version and the current output and range of all channels
syn	Sets or reads the configuration of one of the SYNC outputs
synA	Configures the sync. port A Output (requires sync-board option)
synB	Configures the sync. port B Output (requires sync-board option)
tem	Reads the temperature from one of the three sensors of a selected board in the unit
trig	Trigger command for starting armed generators
vcal	Sets or reads voltage calibration constants
ver	Turns verbose mode on or off
version	Reports firmware version information
vol	Sets or reads the voltage range for a selected channel
wav	Sets or reads which waveform generator is connected to the selected channel or if it is in DC mode

4 Controlling the QDAC Using Python

For communicating through the Python programming language it is necessary to install a module called `pyserial` which facilitates the serial communication. So please install this using the pip command "`pip install pyserial`" or using the install features in your Python integrated development environment (IDE).

The QDevil Python library and QCoDeS and Labber drivers and documentation are available for download from this QR code link:

4.1 Example code

Example python code for sending commands and receiving response using the direct command set:

```
import serial
class qdac():
    def __init__(self, port):
        self.port = port
    def __enter__(self):
        self.sport = serial.Serial(port=self.port,
                                   baudrate=460800,
                                   bytesize=serial.EIGHTBITS,
                                   parity=serial.PARITY_NONE,
                                   stopbits=serial.STOPBITS_ONE,
                                   timeout=0.5)
        return self

    def __exit__(self, type, value, traceback):
        self.sport.close()

    def sendReceive(self, msg):
        self.sport.write(msg + b"\n")
        reply = self.readLine()
        return reply

    def readLine(self):
        out = b""
        c = b""
        while True:
            c = self.sport.read(1)
            if c and c != b"\n":
                out += c
            else:
                break # Timeout or newline
        return out

    def setVoltage(self, channel, volts):
        self.sendReceive(b"set %d %f" % (channel, volts))

with qdac('COM6') as q:
    q.setVoltage(1, 0.05)
```

4.2 Python code delivered with the instrument

QDevil has developed a library of Python methods which interfaces with the direct QDAC commands. The python library is provided by email or download. The required module called `pyserial` facilitating serial communication is not included and needs to be downloaded and installed separately.

QDevil distributes these two Python source files:

`qdac.py`	Implements a class “qdac” which has methods for all the direct commands, including some additional range checks which the QDAC unit itself does not have. This file also serves as documentation.
qdacexample.py	Example of use of some of the qdac() methods, so it is easy to get started.

The QDevil Python code has primarily been tested on both Python 2.7 and Python 3.7.

A list of the methods in the QDevil Python library, qdac.py, is given in Appendix C. See also descriptions in the description of USB/srial commands in Appendix E.

Example Python code using the qdac.py library:

```
import qdac


with qdac.qdac('/dev/tty.usbserial-FTA634UE') as q:
    print("QDAC Serial number: %s" % q.getSerialNumber())
    print("Number of channels: %d" % q.getNumberOfChannels())

    print("-----------------------------------------------")
    print("Setting Channel 1 DC voltage to 1.23456 V")
    result = q.setDCVoltage(channel=1, volts=1.23456)
    print("Result: %s" % result)
    voltage1 = q.getDCVoltage(1)
    print("Voltage output on channel 1 is %f V" % voltage1)
    current1 = q.getCurrentReading(1)
    print("Current on channel 1 is %e A" % current1)

    print("-----------------------------------------------")
    print("Defines triangle function generator for generator 1 and starts it on channel 2")
    result = q.defineFunctionGenerator(generator=qdac.Generator.generator1,
    waveform=qdac.Waveform.triangle, period=100, dutycycle=50)
    print("Result: %s" % result)
    q.setChannelOutput(channel=2, generator=qdac.Generator.generator1, amplitude=1.0,
    offset=0.0)
```

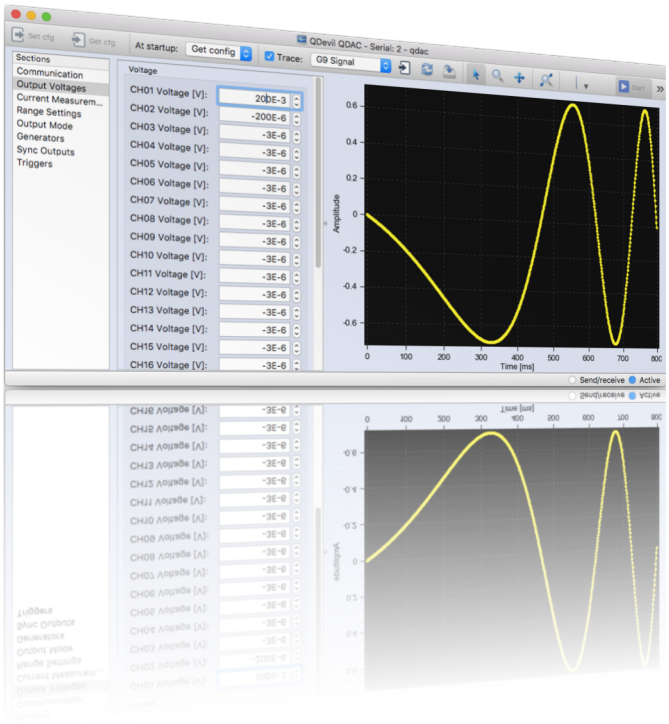
4.3 QCoDeS

QCoDeS is a Python based modular acquisition framework for experimental work with quantum computing devices. It is developed and maintained by the Copenhagen-Delft-Sydney-Microsoft quantum computing program and is open source. A driver for the QDAC is part of the QCoDeS installation (../qcodes/instrument_drivers/QDevil/QDevil_QDAC.py). It is also available when downloading QCoDeS from GitHub. A small introduction is found following the QR code link above.

You find QCoDeS here: https://github.com/QCoDeS

4.4 Labber

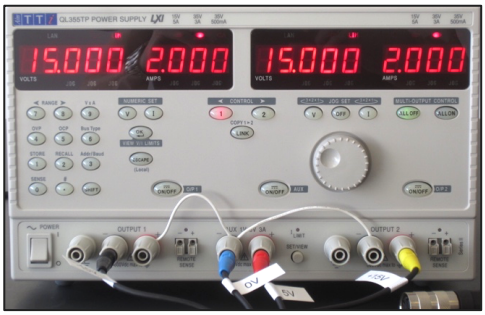
A driver for the Python based laboratory instrument control software, Labber, (see www.labber.org) is part of the Labber distribution. It can also be downloaded and from https://github.com/Labber-software/Drivers. A separate manual is available for this driver. It s found following the QR code link above.



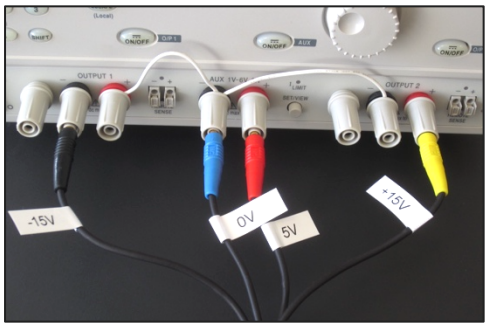
Appendix A Procedure for turning ON the QDAC

> **WARNING**
>
> The QDAC instrument is a sensitive laboratory apparatus. Always follow this checklist before turning on power, and carefully read the manual. The instrument may be damaged if the start-up checklist is not followed.

1. If the power supply has not yet been configured, you must first follow the checklist for configuring the power supply (next page).
2. Turn on the QL355TP power supply
3. Check that the ALL OFF button is lit.
4. Check that Output 1 and Output 2 have been set to 15.000V and 2.000A.



5. Press the SET/VIEW button and check that the AUX has been set to 5.50V and 3.00A.



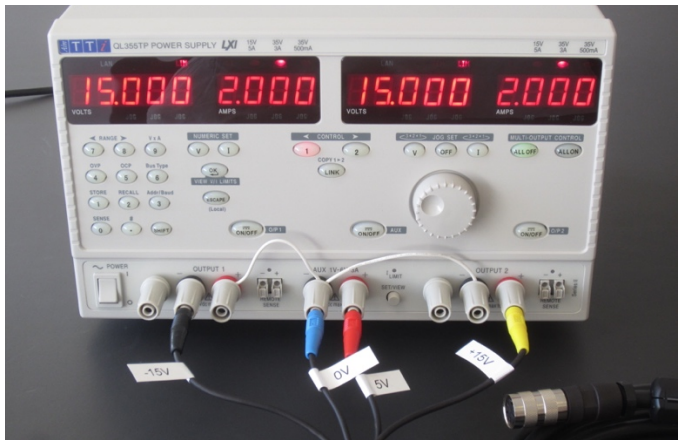
6. Verify that the power wires (**and 0V jumper wires)** are mounted firmly as shown above.
7. Verify that the power cord is firmly inserted into the back of the QDAC. Never disconnect or re-mount power cords unless the **ALL OFF** button is lit.
8. Press the **ALL ON** button (always allow at least 2 seconds before power-on after power-off).

   If you are **not** using a power supply which can switch all voltages on simultaneously, then first switch on 5.5V, then +15V and within 1-2 secs after -15V, see Appendix C.
9. Notice that the LED on the front panel of the QDAC is blinking. For the first 10 seconds during initialization it will blink red-green, then it should start blinking green only (idle mode).
10. To switch off the QDAC unit, press the **ALL OFF** button on the power supply **before** disconnecting the plug from the back side of the QDAC.

Appendix B Procedure for configuring the QL355TP power supply

1. Before starting, ensure that the QDAC prototype is NOT connected to the power supply. Then switch on the power supply using its POWER button.
2. Establish a common reference by connecting "+" on OUTPUT1 to "-" on AUX and to "-" on OUTPUT2. Use a jumper wire with conductor area ~1mm^2 (not included).
3. Set Output 1 and Output 2 to 15.000V and a 1.500A current limit. For 48-chan systems set the limiter to 2.000A.
4. Press the SET/VIEW button and set the AUX output to 5.50V and a 3.00A current limit.
5. Press SHIFT # 33 on the keyboard. This will lock the settings of the power supply.
6. Verify that the output voltages and currents cannot be changed when turning the knobs of the power supply.
7. Check that the ALL OFF button is lit.
8. Connect all 4 bananas from the QDAC power cord to the power supply outlets as shown below. Note the voltages printed on the flags on the wires:

Black 	-15 V	OUTPUT1 -
Blue 	0 V (Common)	AUX -
Red 	+5.5 V	AUX +
Yellow 	+ 15 V	OUTPUT2 +



9. Ensure that the cables are firmly mounted, so that they will not fall off.
10. Connect the power cord to the backside of the QDAC and lock the connector by turning its shield clock-wise so that the plug goes all the way into the socket and so that it will not fall off easily.
11. Follow the instructions on the previous page to switch on the QDAC unit.

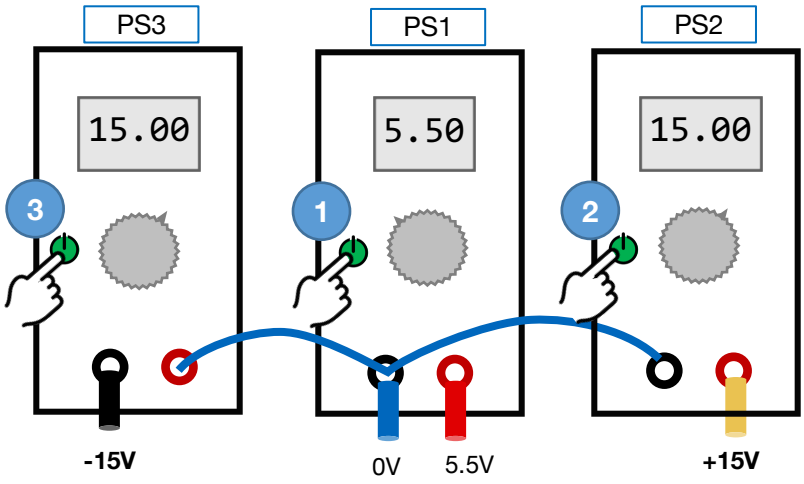
If you have difficulties setting up the power supply or the QDAC unit, or if you have other questions, please contact QDevil at info@qdevil.com.

Appendix C Using separate supplies (not recommended)

It is recommended to us a single power supply capable of supplying all voltages at the same time, such as the QDevil Power Supply or an advanced laboratory power supply like the QL355TP or a Keysight E36313A. If this for some reason is not possible then please follow this checklist for using three separate supplies:

Make sure that the outputs of the 3 power supplies are galvanically isolated from mains and from each other.

1. Set PS1 to 5.5V, PS2 to 15V and PS3 to 15V.
2. Set the 5.5V current limiter to 3A and the 15V current limiters to 1.5A (24ch units) or 3A (48ch units).
3. Switch all power supplies off (assuming that they will remember their set voltages at power-on)
4. Connect 0 on PS1 to the hot terminal (15V) on PS3 and to 0 on PS2
5. Connect the black banana plug to 0 on PS3
6. Connect the blue banana plug to 0 on PS1 (also connected to +15V on PS3 and 0 on PS2)
7. Connect the red banana plug to the hot terminal (5.5V) on PS1
8. Connect the yellow banana to the hot terminal (15V) on PS2
9. Now switch on PS1
10. Then switch on PS2 (+15V)
11. Within 2 seconds after switch on PS3 (-15V). The order is **very** important.



Schematic of how to wire up three separate power supplies, with the common 0V reference established by the wire drawn in blue (VERY IMPORTANT). The power up sequence is also indicated. It is very important that +15V is switched on *before* -15V, but also that the delay between them is maximum a couple of seconds.

Appendix D Alphabetical list of python methods in qdac.py

The methods in qdac.py responds with either nothing, a single value (in rare cases a string), or multiple values. In case of multi valued replies a python dictionary is returned.

Qdac.py contains a number of enumerating classes to use with the Python methods for your convenience:

For generators: *Generator.generator1, ... , Generator.generator8, Generator.pulsetrain, Generator.AWG*.

For waveforms: Waveform.*sine, Waveform.square, Waveform.triangle, Waveform.staircase.*

As new firmware versions are released also new qdac.py libraries are released. Therefore the qdac.py has a version number.

List of methods in qdac.py version 1.21.

Python method	Return type	Related Command
defineAWG(samples, repetitions, trigger) *Programs and starts/arms the AWG (generator no. 9) with the list/array of voltages given by samples. (max size 8000).*	Dict	**awg**
defineFunctionGenerator(generator, waveform, period, dutycycle, repetitions, trigger) *Sets the waveform, curve details, and number of repetitions for one of the eight (1-8) generators using predefined waveforms and starts or arms the generator. Note: 'period' and 'dutycycle' have different meaning for the four different waveforms: For sine, 'dutycycle' is obsolete, for triangle and square 'dutycycle' is actually the duty-cycle, for staircase 'period' is the step length and 'dutycycle' the number of steps.*	Dict	**fun**
definePulsetrain(lowDuration, highDuration, lowVolts, highVolts, repetitions, trigger) *Sets the durations (millisec) and levels (Volts) of the high and low segments of the pulse train generator (no. 10) and the number of pulses (-1 = infinite).*	Dict	**pul**
executeTrigger(triggerNumber) *Fires a trigger (1..10) which starts one or more generators.*	Str	**trig**
getAwg() *Returns the configuration of the arbitrary waveform generator (awg), except for the actual array of samples.*	Dict	**run**
getCalibrationChannel() *Returns which channel is currently connected to the calibration output BNC.*	Val	**cal**
getChannelOutput(channel) *Returns the mode (dc=0, generator=1-10), and the amplitude and offset of any generator connected to a DAC channel.*	Dict	(multiple)
getCurrentCalibration(channel, theRange) *Reads out the current sensor calibration parameters from a channel for the given range (theRange = 1e-6 or 1e-4).*	Dict	ical
getCurrentRange(channel) *Returns the current sensing range for a DAC channel. The range is return in units of Amperes as 1e-6 or 1e-4.*	val	cur
getCurrentReading(channel) *Reads current from a DAC channel. Unit is in Amperes.*	Val	get

Author	Date	Document no
AK	27-04-2021	PM10091-14

getDCVoltage(channel) *Reads the DC voltage from a DAC channel. Please make sure the channel is in DC mode.*	Val	set
getExternalSampleClock() *Reports if an external sample clock (sync port A in) is being used: 0=no, 1= yes. Requires the sync. board option.*	Val	extclk
getFunctionGenerator(generator) *Reads out the type, period, dutycycle, and number of repetitions of a generator.*	Dict	fun
getMinMaxOutputVoltage(channel, theRange) *Reads out the minimum and maximum output voltages a channel (1-24) in the given range (theRange = 1 or 10).*	Dict	rang
getNumberOfChannels () *Returns the number channels in the connected unit*	Val	-
getPulsetrain() *Returns the configuration of the pulse train generator.*	Dict	pul
getRawCurrentADCreading(channel) *Reads current in raw 24 bit digital ADC units from a channel.*	Val	adc
getRawDAC(channel) *Reads the currently set 20 bit digital DAC value (-524288 to 524287) units from a channel.*	Val	dac
getSerialNumber() *Returns the serial number of the QDAC. See 'sernum' command.*	Str	sernum
getSyncOutput(syncChannel) *Returns the output state from a sync channel (0-5): which generator it is bound to, delay, duration, and pulse length.*	Dict	syn
getSyncPortOut(port) *Reports the configuration of sync ports A or B (port = 'A' or 'B'). Requires the sync. board option.*	Val	synA synB
getVoltageCalibration(channel, theRange) *Reads out the voltage calibration parameters for a channel for the given range (theRange = 1 or 10).*	Dict	vcal
getVoltageRange(channel) *Returns the voltage output range of a QDAC channel. Range is returned as 1.0 or 10.0 for the 1.1V range and the 10V range, respectively.*	Val	vol
getCurrentRange(channel) *Gets the current measurement range (1e-6 or 1e-4) of the selected channel. The effective measurement ranges are 0.75e-6A and 75e-6A ranges, respectively.*	Val	cur
linkExternalTrigger(triggerNumber) *Tells a trigger (1..10) to fire when a pulse appear on the trigger in (synB input) port. Requires sync. board option.*	Str	trig
readTemperature(board, position) *Reads the temperature in degrees Celsius of one of the DAC boards (0-5, top board is 0). Sensor number 0 is at the right end of the board (near ch. 8, 16, 24 ..), sensor 1 in the middle, and sensor 2 in the left end.*	Val	tem
restart() *Reboots the QDAC (restarts the firmware).*	-	-

setBaudrate(baudrate) *Sets the USB / serial communication rate of the QDAC.*	-	brt
setCalibrationChannel(channel) *Sets which channel should be connected to the calibration output BNC.*	Val	cal
setChannelOutput(channel, generator, amplitude, offset) *Sets the mode (dc=0 or a generator=1-10) of a channel, and in generator mode the amplitude and offset. For the AWG (9) and the pulse train generator (10) the <amplitude> is a scaling factor whereas for the 1-8 fixed waveform generators both <amplitude> and <offset> are both actual voltages.*	-	wav
setCurrentCalibration(theRange, microampsPerSample, offsetMicroamps) *Sets the current calibration parameters for the current sensor A/D converter of a channel for the given range (theRange = 1e-6 or 1e-4).*	-	ical
setCurrentRange(channel, theRange) *Sets the current measurement range (theRange = 1e-6 or 1e-4) of the selected channel. The effective measurement ranges are 0.75µA and 75µA ranges, respectively.*	Val	cur
setDCVoltage(channel, volts) *Sets the DC voltage of a DAC channel. This only works if setChannelOutput has been set to Generator.DC.*	Dict	set
setExternalSampleClock(self, enabledisable) *When <enabledisable> = true an external sample clock (nominally 1KHz) applied on Sync port A is used.*	Str	extclk
setRawDAC(dacValue) *Sets the DAC output of a channel using the raw 20 bit digital DAC value (dacValue = -524288 to 524287).*	Dict	dac
setSyncOutput(syncChannel, generator, delay, pulseLength) *Set which generator should give a pulse of given length (millisec) on which sync output after a specified delay (millisec).*	Dict	syn
setSyncOutputOff(syncChannel) *Disable sync output on the specified sync channel.*	Dict	syn
setVoltageCalibration(channel, theRange, samplesPerVolt, offsetSamples) *Sets the voltage calibration parameters for a channel for the given range (theRange = 1 or 10). Note that the parameters are not persisted by this command.*	-	vcal
setVoltageRange(channel, theRange) *Sets the current measurement range (theRange = 1 or 10) of the selected channel, for the 1.1V and 10V ranges, respectively.*	Val	vol
storeCalibrationInFlash() *Stores the current (modified) calibration parameters for all channels, all voltage and current ranges, in flash memory so that they are not erased at power off.*	Val	savecalib5599
syncPortOut(port, enabledisable) *Enables or disables sync ports A or B (port = 'A' or 'B'), for outputting the sample clock (port A) or trigger signal (port B), respectively. Requires the sync. board option.*	-	synA synB
waitForSync(generator, timeout) *Waits and returns when the specified waveform generator restarts. 'timeout' specifies the maximum number of seconds to wait, '-1' means infinite.*	Val	ssy

Appendix E Alphabetical list of USB/serial commands

This list applies for firmware version 1.06.

adc Get raw digital current sensor ADC reading

get <channel>

channel = 1 – 24

Purpose:

For diagnostics and calibration. Measures the output current of a given channel and returns a value in the 24 bit range (-8388608 : 8388607) of the AD converter. During conversion time, ongoing tasks will continue, while new will await ADC conversion and be put into queue. Returns result after a delay of 200ms, which relates to the ADC conversion time. Note that when the current is changed (by changing the voltage output), the ADC needs some time to settle 100-200ms.

The trusted ranges are slightly smaller than the nominal ranges. Outside the practical ranges the ADC may go into saturation.

The sensitivity depends on the current range set by the “cur” command:

cur	Nominal Range	Digit range	Practical range
0	±1 µA	-8388608 : 8388607	±0.75 µA
1	±100 µA	-8388608 : 8388607	±75 µA

Returns:

‘ <current>\n’ (verbose off) or

'*Channel <channel> rawcurrent: <adcdigits>* \n' (verbose on)

awg Arbitrary Waveform Generator

awg <type> <interval> <sample point data>

type = 0 (sample data), [Options 1 and 2 are deprecated]

[interval = 1 – 268435455 (ms) - Only used for deprecated spline interpolation]

'type' and 'interval' values will be ignored, but shall be present, in subsequent commands when data are split between commands.

sample point data =

> Sample data consist of contiguous floating point value sample data. When executing the *wav* command the sample point data will be scaled and offset into output Voltages.
>
> Max 64 pr. command.

> Data can be split between several commands, or needs to because of the limited number of sample point data per command
> The sent data will be treated/calculated/output after the receipt of a 'run' command, which is always required

Purpose:

The Arbitrary Waveform Generator can be used to output a customized waveform uploaded to the QDAC. The waveform can be up to 8000 milliseconds long. In raw mode (the only mode after deprecating other functions), data must be uploaded representing each one millisecond output sample.

The generator is numbered 9.

returns:

'\n' (verbose on or off) (see 'run' command for verbose feedback)

brt Select baudrate

brt <baudrate>

baudrate = preferably a standard rate, though the QDAC will try to use any given

Alternatively, a number between 2 and 14 can be sent (see below).

Either way it is strongly advised to use highest possible rate below 921600 (experimental), as low rates can impair functionality.

The values gives:

2: 600, 3: 1200, 4: 2400, 5: 4800, 6: 9600, 7: 14400, 8: 19200, 9: 38400, 10: 57600, 11: 115200, 12: 230400, 13: 460800, 14: 921600

Purpose:

Used to change the communication baudrate of the equipment.

The default at power on is 460800.

Returns:

'\n' (verbose off) or

'***Changing BAUD rate to: <baudrate>\n***' (verbose on).

(The feedback is sent in the old baudrate, before changing.)

cal Select channel to calibration connector

cal [<channel >]

channel = 1 – 24 (0 for none)

Purpose:

Routes the output of a selected channel to the calibration connector.

If channel number is omitted the currently selected channel number is returned.

The default at power on is 0 for none.

Returns:

'\n' (verbose off) or

'***Channel selected: <channel number>\n***' (verbose on).

If query:

'***<channel number>\n***' (verbose off) or

'***Channel selected: <channel number>\n***' (verbose on).

cur Set ADC current range of the selected channel

cur <channel> [<on/off>]

channel = 1-24

on/off = 1/0 (0: low = 1μA, and 1: high = 100μA)

Purpose:

Sets the current range for the given channel.

If only channel number is given, state is returned.

The default at power on is 1, low (100μA).

> Note, it the Voltage range (“vol”) is set to 1 (±1.1V) then the current sensing range can only be 1μA (0), whereas it can be both 1μA (0) and 100μA (1) when the channel is in the 10V range.

Returns:

‘\n’ (verbose off) or

‘Current range on Channel <channel> set to: <range>\n’ (verbose on) (range is ‘Low’ or ‘High’)

If query:

'Current range on Channel <channel> set to: <range>\n’ (verbose on) (range is ‘Low’ or ‘High’) or

'<range>\n’ (range is ‘0’ or ‘1’) (verbose off)

Author	Date	Document no
AK	27-04-2021	PM10091-14

dac Set or read the raw digital input to the DC output voltage of DAC channel

dac <channel> [<raw_val>]

channel = 1 – 24

raw_val = -524288 to 524287

If voltage is omitted the currently output voltage on the channel number is returned.

Note

Does not put the channel into DC mode if it's in a waveform mode. DC is selected as default signal source at power on and is identified as function generator number 0.

Returns:

'\n' (verbose off)

'Digital output: <raw_val> on Channel: <channel>\n' (verbose on)

If query:

'<raw_val>\n' (verbose off)

'Channel: <tab> <channel> <tab> <raw_val>\n (verbose on)

extclk Use external sample clock

extclk [<option>]

option = 0: disable (default), 1: use clock from the Sync-A input

Controls if the QDAC should use its internal 1 kHz sample clock or should use the incoming clock on the synA input port on the rear panel of the QDAC, if this option is present. To synchronize the sample output to another QDAC (outputting its sample clock) set this option to 1 (first connect the synA input to the synA output on the other QDAC!)

Returns:

'\n' (verbose off) or

'*External Clock: <option>\n*' (verbose on)

If query:

'*External Clock: <option>\n*' (verbose on)

or

'*<option>\n*' (verbose off)

fun Function Generator

fun <generator> [<waveform=1> <period> [<repetitions>] [<trigger>]]

fun <generator> [<waveform=2,3> <period> [<duty cycle>] [<repetitions>] [<trigger>]]

fun <generator> [<waveform=4> <step length> <no. of steps> [<repetitions>] [<trigger>]]

generator	=	1 – 8
waveform	=	1 (sine), 2 (square), 3 (triangle/ramp), 4 (staircase)
period	=	Waveform period in milliseconds (waveforms 1-3).
duty cycle	=	0-100* (waveforms 2-3)
step length	=	step length in milliseconds (waveform 4).
no. of steps	=	1.. 268435455 (for waveform 4, staircase).
repetitions	=	0 – 268435455, or -1 (-1 = endlessly, default). 0 will stop the generator,
trigger	=	1-10 specifies an optional trigger, which is used to start the generator. If a trigger is not specified (or is zero) the generator will start immediately.

*) For triangle, a duty cycle of 0, gives falling ramp and 100 gives rising ramp.

See the comment in the description of the ‘wav’ command regarding the resulting signal output

Purpose:

If only the generator number is given (query), the active setup of the generator is returned.

Returns:

‘\n’ (verbose off) or

'Curvetype: 1, Period: <period>, Repetitions: <repetitions>, Trigger:<trigger>\n' (verbose on) or

'Curvetype: 2, Period: <period>, Dutycycle: <duty cycle>, Repetitions: <repetitions>, Trigger:<trigger>\n' (verbose on) or

'Curvetype: 3, Period: <period>, Dutycycle: <duty cycle>, Repetitions: <repetitions>, Trigger:<trigger>\n' (verbose on) or

‘Curvetype: 4, Step length: <steplength>, No. of steps: <no. of steps>, Repetitions: <repetitions>, Trigger: 10 (verbose on)

If query:

Verbose on: Same as verbose response above

Verbose off:

'1, <period>, <repetitions>, <remaining no of repetitions>, <trigger>\n' (verbose off) or

'2, <period>, <duty cycle>, <repetitions>, <remaining no of repetitions>, <trigger>\n' (verbose off) or

'3, <period>, <duty cycle>, <repetitions>, <remaining no of repetitions>, <trigger>\n' (verbose off) or

'4, <step length>, <no. of steps>>, <repetitions>, <trigger>\n' (verbose off)

Note that the remaining number of repetitions is not reported for the staircase generator (4) in firmware version 1.07.

get Get current sensor ADC reading

get <channel>

channel = 1 – 24

Purpose:

Measures the output current of a given channel and returns a value representing μA. During conversion time, ongoing tasks will continue, while new will await ADC conversion and be put into queue. Returns result after a delay of 200ms, which relates to the ADC integration time. Note that there is a dead time between consecutive readouts of about 150 ms, and that the current sensor also needs 100-200 ms to settle when the current is changed (by changing the voltage)

The sensitivity depends on the current range set by the “cur” command:

cur	Range	Practical range
0	±1 μA	±0.75 μA
1	±100 μA	±75 μA

Returns:

‘ <current>\n’ (verbose off) or

'Channel <channel> current: <current> uA\n' (verbose on)

ical Sets or reads the current calibration parameters of the selected channel/range

ical <channel > <range > < microampsPerSample > < offsetMicroamps >

channel = 1 – 24

range = 1/0 (0: ±1µA and 1: ±100µA, nominal ranges)

microampsPerSample = micro amperes per raw ADC sample (digital value)

offsetMicroamps = actual current when the AD converter reports a digital value of zero

Purpose:

Each channel has individual linear calibration constants for its current sensor in both ranges. With this command, new calibration constants can be set, or the existing ones can be read out.

In order to persist new calibration parameters between power down and ups, the parameters can be saved to flash memory using the savecalib5599 command.

The measured current in micro Amperes is calculated as:

Current(microAmps) = raw_value x microampsPerSample + offsetMicroamps

Example (write, 1µA range):

ical 12 0 81.34e-9 25.567

Returns:

'\n' (verbose on or off)

If query:

'Calibration for ch: <channel> for range <rangeinµA> uA mult 8.12544e-08 offset 2.761134e-01\n' (verbose on)

pul Put pulse train on output

pul [<low time> <high time> <low value> <high value> [<pulse count>] [trigger]]

low time = 0 – 2147483647 (milliseconds)

high time = 0 – 2147483647 (milliseconds)

low value = -10.000000 – 10.000000 (Volts*)

high value = -10.000000 – 10.000000 (Volts*)

pulse count = 1 – 268435455 (optional – omitted or -1 means forever until new command)

*) The low/high values may be scaled and offset when executing the *wav* command.

Purpose:

Gives a pulse train of 1 - ∞ pulses. The low and high times can be set individually. The same is the case for the low and high voltage value.

If no arguments given, the setup will be returned. When relevant, a returned pulse count of -1, means forever. The generator is numbered 10.

Returns:

'\n' (verbose off) or

'Pulsetrain, low time: <low time>ms, low value <low value>V, high time: <high time>ms, high value: <high value>V, <pulse count> times, trigger: <trigger>\n' (verbose on)

If query:

Or

'Pulsetrain, low time: <low time>ms, low value <low value>V, high time: <high time>ms, high value: <high value>V, <pulse count> times, trigger: <trigger>\n' (verbose on) or

'<low time>, <low value>, <high time>, <high value>, <pulse count>, <trigger>\n' (verbose off)

rang returns the output range of the specified channel

rang <channel> <range>

channel = 1 - 24

range = 1/0 (0: ±10V and 1: ±1.1V)

Purpose:

Displays the calibrated minimum and maximum output voltages of the specified channel for the specified range..

Returns:

'For 10V range on channel **<channel>*, MIN: -10.000272 MAX: 9.999519\n'*** (verbose on or off)

run Complete AWG generation

run [<repetitions>] [<trigger>]

repetitions = -1 – 2^{31}-1 (-1 = endlessly)

trigger 1-10 specifies and optional trigger, which is used to start the generator. If a trigger is not defined the awg will start immediately (referred to as trigger no. 0)

Purpose:

Completes the generation of the arbitrary waveform defined using the 'awg' command one or more times.

Number of repetitions of the waveform (cycles) defaults to -1 (endlessly).

The waveform can be repeated by a new run command, without having to reload.

returns:

'\n' (verbose off) or

'Waveform of type 0 with <number of samples> samples, <repetitions> repetitions, <trigger no> trigger \n' (verbose on)

If query ('run'):

'Waveform of type 0 with <number of samples> samples, <repetitions> repetitions, <trigger no> trigger \n'

savecalib5599 Stores the present calibration parameters in flash

savecalib5599

Purpose:

Stores the current (modified) calibration parameters for all channels, all voltage and current ranges, in flash memory so that they are not erased at power off,

returns:

'DATA SAVED\n' (verbose on/off)

set Set the DC output voltage of DAC channel

set <channel> [<voltage>]

channel = 1 – 24

voltage = -1.1V to +1.1V or -10V to +10V (nominally*), depending on the voltage range

It is, as the only signal source, unaffected by 'amplitude' and 'offset' set by **wav** command.

DC is selected as default signal source at power on and is identified as function generator number 0.

If voltage is omitted the currently output voltage on the channel number is returned.

*) Due to the tolerances of components the limits for the accessible voltages are -10±0.002V to +10±0.002V for the 10V range and -1.1111±0.0004V to +1.1111±0.0004V for the 1.1V range, respectively. Trying to set the voltage outside the calibrated limits for the current voltage range will generate an error.

> **Note**
> Does not put the channel into DC mode if it's in a waveform mode. Use the "wav" command for that (at startup all channels are in DC mode).

Returns:

'\n' (verbose off)

'Output: <voltage> (<raw_val>) on Channel: <channel>\n' (verbose on)

If query:

'*<voltage>\n'* (verbose off)

'Output: <voltage> to Channel: <channel>\n' (verbose on)

SSY Soft synchronization pulse sent as data

ssy <func.gen.>

func.gen. = 1 – 10

Purpose:

Will send the number of the selected source at the start of the next period (end of present) to the host computer. Will run only once.

Returns:

'\n' (verbose off) or

'***Soft sync from generator: <func.gen>\n***' (verbose on) right away, then:

'#<func.gen>\n' at the start of the next waveform period (end of present), where <func.gen.> is form of '01' – '10'

status List status of all channels

status

Purpose:

For finding the current version of the firmware and the output setting and current sensing mode for each channel.

Returns:

The firmware version followed by the output voltage, voltage range and current sensing range of each channel.

Returns

```
Software Version: 1.02
Channel Out V           Voltage range   Current range

 1        0.000000              X 1             hi cur
 2        0.000000              X 1             hi cur
 3        0.000000              X 1             hi cur
 4        0.000000              X 1             hi cur
 5        0.000000              X 1             hi cur
 6        0.000000              X 1             hi cur
 7        0.000000              X 1             hi cur
 8        0.000000              X 1             hi cur
..
```

Spaces are tab characters and there is a new line character at the end of each line

syn Synchronization pulse on sync output

syn <number> [<func.gen.> <delay> <pulse length>]

number = 1 – 2 for 24 ch units (and 1-5 for 48 ch units)

func.gen. = 1 – 10, 0 means turned of.

delay = 0 – 268435455 milliseconds (delay of between n and n+1 periods means you will only get a sync pulse every n+1 periods because it will not retrigger until the first pulse has started)

pulse length = milliseconds

Purpose:

Generates galvanic isolated 5V pulses with a set dependency of one of the signal sources, on one of the synchronization outputs.

Repeats forever or until source (func.gen.) set to 0.

If only generator number is given, active setup of the sync output is returned.

Returns:

'\n' (verbose off) or

'***Sync output <number>, bound to generator <func.gen.>, delay <delay>ms, duration <pulse length>\n***' (verbose on)

If query:

'<func.gen.>, <delay>ms, <pulse length>\n' (verbose off)

or

'Sync output <number>, bound to generator <func.gen.>, delay <delay>ms, duration <pulse length>\n' (verbose on)

synA Configures sync. port A Output

synA [<option>]

option = 0: disable (default), 1: output sample clock

Controls what is output on the SynA output port on the rear panel of the QDAC , when this option is present. To output the sample clock for synchronizing other QDAC units to the current unit, option 1 should be chosen. This may also be used to trigger other instrumentation which has to be in pace with the QDAC.

Returns:

'\n' (verbose off) or

'***SynA: <option>\n***' (verbose on)

If query:

'***synA: <option>\n***' (verbose on)

or

'***<option>\n***' (verbose off)

synB Configures sync. port B Output

synB [<option>]

option = 0: disable (default), 1: output generator triggers

Controls what is output on the SynB output port on the rear panel of the QDAC, when this option is present. To output a pulse every time the *trig* command is executed, choose option 1. This is used for example for synchronizing generators in multiple QDACs.

Returns:

'\n' (verbose off) or

'***synB: <option>\n***' (verbose on)

If query:

'***synB: <option>\n***' (verbose on)

or

'***<option>\n***' (verbose off)

tem Read board temperature

tem <board> <sens. number>

Board = 0 – 5 (board 0 is ch. 1-8, board 1 is ch. 9-16, etc)

Sens. Number = 0 – 2

Purpose:

Reads the temperature in degrees Celsius, at one of three physical places at the selected board. Sensor number 0 is at the right end of the board and sensor 2 is in the left end of the board, sensor 1 in the middle approximately.

Returns:

‘ <temperature>\n’ (verbose off)

or

‘Board <board> Sensor <sensor>: <temperature>\n’ (verbose on)

trig starts armed generators

trig <trigger number> [<sync link>]

trigger number = 1 – 10

sync link = 1: fire on pulse at Sync-B input,

0: disconnects all triggers from the external Sync-B In input

Purpose:

Fires the specified trigger number which will start any idle waveform generators set up to use that particular trigger.

If <sync link> is specified and equals 1, the trigger will fire next time there is a trigger pulse on the trigger input Sync-B In, on the rear of the QDAC cabinet. Sync link only works for QDACs with the optional synchronization ports on the rear panel.

Returns:

'***SynB-in connected to trigger: <trigger number>\n***' (verbose on or off)

If query ('trig'):

Same as response

vcal Sets or reads the voltage calibration parameters of the selected channel/range

vcal <channel > <range > < samplesPerVolt > <OffsetSamples>

channel = 1 – 24

range = 1/0 (0: ±10V and 1: ±1.1V)

samplesPerVolt = number of raw DAC samples (digital values) per volt

OffsetSamples = raw (digital) DAC value for zero output voltage

Purpose:

Each channel has individual linear calibration constants for its voltage output in both ranges. With this command, new calibration constants can be set, or the existing ones can be read out.

In order to persist new calibration parameters between power down and ups, the parameters can be saved to flash memory using the savecalib5599 command. Note that there is no command for restoring factory settings.

The raw DAC digital value is calculated as:

raw_value = requested_voltage x samplesPerVolt + OffsetSamples

Returns:

'\n' (verbose on or off)

If query:

'Calibration for ch: <channel> for range <range in Volts> mult < samplesPerVolt > offset < OffsetSamples >\n' (verbose on)

or

<channel> <range in Volts> < samplesPerVolt > < OffsetSamples>\n' (verbose off)

ver Turn verbose mode on or off

ver <on/off>

on/off = 1 – 0 (default 1)

Purpose:

Controls whether all commands return data or not, and whether get commands return just the number or number plus label. With verbose off, all commands at least return '\n'

The default at power on is on.

returns:

'\n' (verbose on or off)

version Reports firmware version information

version [<option>]

option = omitted: Version number, 1: Build number, 2: Commit ID, 3: Build date, 4: Build time

Purpose:

Makes it possible for driver software to determine the exact firmware version in order to check that the firmware is compatible with the driver.

Returns: (verbose on or off)

>version
Software Version: 1.06

>version 1
Build: 1000208

>version 2
Commit: 254656

>version 3
Date: Apr 25 2019

>version 4
Time: 14:15:55 '\n'

vol Set DAC voltage range on a selected channel

vol < channel > <range>

channel = 1 - 24

range = 1/0 (0: ±10V and 1: ±1.1V)

Purpose:

Sets the voltage range for the given DAC channel.

If only channel number is given, the state is returned.

Default state is 0 (±10V).

When the <range> is set to 1 (1.1V range) the current measurement range will automatically be switched to the ±1µA range in order ensure equal power consumption (hence heating) in the internal relays of the QDAC.

Warning

It is advised not to change the voltage range of a channel while it is connected to an experiment, at least not without first setting the voltage to zero as unexpected voltage jumps or spikes may occur. See exact behavior on the following page.

Returns:

'**\n'** (verbose off) or

'**Voltage range on Channel <channel> to: <range>\n**' (verbose on) (range is 'X 1' for 10V or 'X 0.1' for 1V)

When changing to the 1.1 V range:

'Voltage range on Channel <channel> set to: X 0.1, current range on 1uA\n' (verbose on)

If query:

'Voltage range on Channel <channel> set to: <range>\n' (verbose on) (range is 'x1' or 'x0.1') or

'<range>\n' (range is '0' or '1') (verbose off)

If in the 1.1 V range:

'Voltage range on Channel <channel> set to: X 0.1, current range on 1uA\n' (verbose on)

'<range>\n') (verbose off)

(vol continued…)

Detailed behaviour when changing the voltage range (firmware versions 1.02-1.07):

When the voltage range is changed from HIGH to LOW:

In DC mode: Output is attenuated 9 times immediately

In waveform mode:

Offset < 1.1V: Voltage stays the same but gives a negative spike to 1/9th of the current voltage

Offset > 1.1V: Voltage is clipped at 1.1V, negative spike to 1/9th of the current voltage

When the voltage range is changed from LOW to HIGH:

In DC mode:

Vset < 1.1V: Output jumps to Vset x 9

Vset > 1.1V but Vread <1.1V: Output jumps back to 9 times the current voltage*

In waveform mode:

Offset < 1.1V: Voltage stays the same but gives a positive spike to 9 times the current voltage

Offset in high range previously set to > 1V: Voltage jumps to offset and gives a spike up to 9 times the current voltage*

* this situation may happen if the voltage range is changed from high to low and then back to high without setting a new DC or offset voltage, respectively.

wav Select waveform on output

wav <channel> [<waveform generator> <amplitude> <offset>]

channel = 1 – 24

waveform generator = 0 (DC), 1 – 8 (sine/square/ramp and triangle func. gen.), 9 (AWG), 10 (pulse generator)

amplitude, offset = -1.1V to +1.1V or -10V to +10V (nominally*), depending on the voltage range

For the AWG (9) and the pulse train generator (10) the <amplitude> is a scaling factor whereas for the 1-8 fixed waveform generators both <amplitude> and <offset> are both actual voltages, however with a slightly different meaning: For the Sine waveform <offset> the signal will swing from -<amplitude> to +<amplitude> around the <offset> level, whereas for the square wave, triangle and staircase waveforms the signal will go from<offset> to <amplitude>. See also ‘fun’ command.

For DC output (0) the <amplitude> and <offset> have no effect.

If only channel number is given, the chosen waveform generator, amplitude and offset is returned.

If the requested combination of generator, amplitude and offsets results in clipping an *out-of-range* error message will be returned. Note that range checking is not performed if one of the generators is changed, using the ‘pul’ or the ‘awg’ (and ‘run’) command, *after* the ‘wav’ command has been executed. In that case the output will be clipped.

*) Due to the tolerances of components the limits for the accessible voltages are -10±0.002V to +10±0.002V for the 10V range and -1.1111±0.0004V to +1.1111±0.0004V for the 1.1V range, respectively. Trying to set the voltage outside the calibrated limits for the current voltage range will generate an error.

Purpose:

Connects a DAC channel to one of the waveform generators and sets the amplitude and offset.

Returns:

‘\n’ (verbose on or off) No verbose output

In case the waveform will be out of range on the selected channel:

‘Error: OutOfRange>\n’’

If query:

'Generator: <n>, Amplitude: <amplitude>, Offset: <offset>\n' (verbose on) or

'**<n>,<amplitude>,<offset>\n**' (verbose off).

Appendix F Special 10mA version (Q303)

In the Q303 special high current, 10mA, version of the QDAC the output circuit for the high current range has been modified to allow currents up to 10mA within specifications, even in resistive loads up to 1kOhm.

The table below highlights the difference between the standard model and the Q303 special version, and how to translate the voltage/current modes of the standard model to those in the high current model.

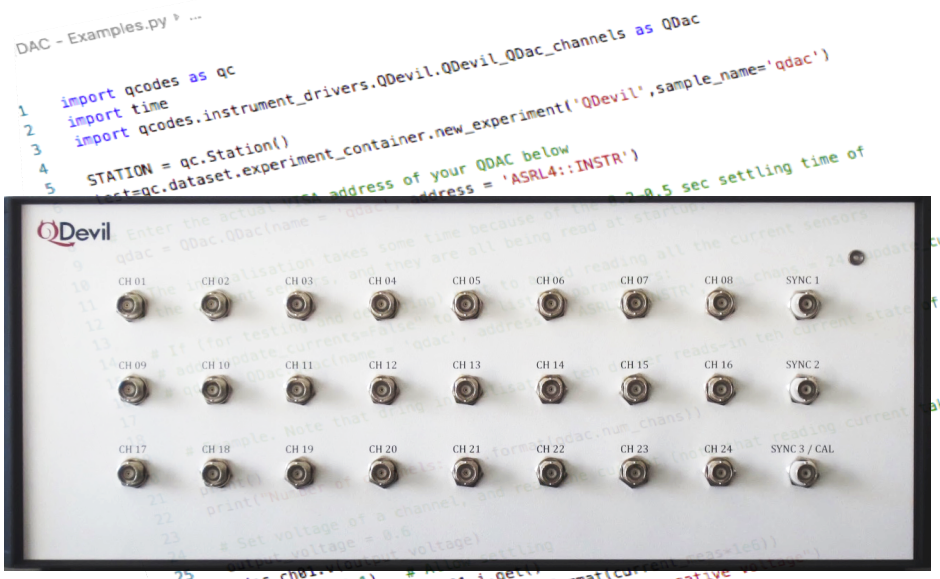
The 10mA capability is in the high voltage (nominally ±10V) combined with the high current sensing mode (nominally ±100 µA). Note that the firmware and drivers do not “know” that it is a high current model, so it is not reflected in the command set. Whenever standard ranges are referred to in the present manual, please translate according to this table:

Voltage range	Nominal current sensing range	Actual current sensing range (approx. resolution)	Guaranteed current sourcing range
±10 V	±100 µA	±75 µA (5 nA)	±1 mA
	±100 µA	±7.5 mA (1 µA)	±10 mA
±10 V	±1 µA	±0.75 µA (0.2 nA)	±10 µA
±1.1 V	±1 µA	±0.75 µA (0.2 nA)	±10 µA

	Q302 Standard version (and Q301 48 channel version)
	Q303 Special 10mA version
	Both versions

Note that the power supply for the 10mA QDAC must be able to supply 1.5 A from its ±15V terminals, where it is 0.8 A for the standard model.

QCoDeS driver for the QDevil QDAC



This driver provides basic methods for setting and ramping the output voltage, reading the current sensors, and setting the voltage output ranges and current measurement ranges of all channels, but also facilitates synchronized multi-channel voltage ramping.

The driver requires a QDAC from QDevil with firmware 1.07 or later installed.

Author: AK/MVS

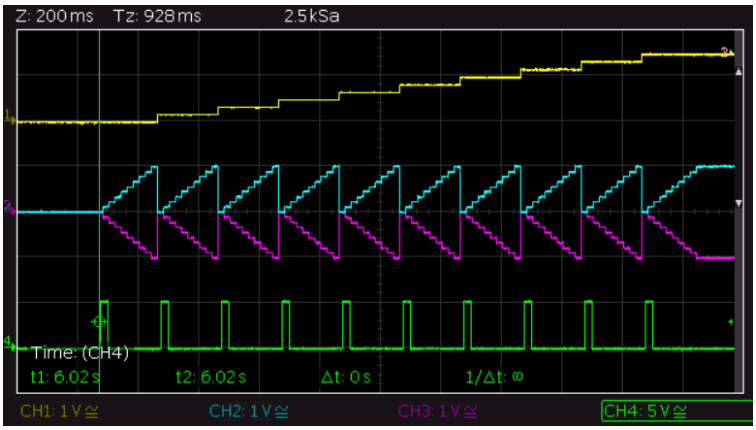
Date: 2020-02-11
Doc ver: 2.1

1 Introduction

It is recommended to consult the QDAC user's manual before attempting to operate the instrument using QCoDeS in order to get an overview of the QDAC's functionality and capabilities.

- This driver consists of a single Python file: QDevil_QDAC.py
- A Jupyter notebook with example usage.

This QCoDeS driver is based on the "QDev\Qdac_channels.py" driver from the qcodes instrument drivers. This driver leverages several of the latest developments and requires at least firmware 1.07 from QDevil, and a QDAC from QDevil. The improvements include auto configuration, output calibration, and synchronized multi-channel ramping.



Example of 2D multi-channel ramping

2 Installing the driver

If your installed copy of QCoDeS includes the QDAC driver (for example from installing pull request 1613) it will be located the "QDevil" subfolder in the "qcodes\instrument_drivers" folder, and you don't need to do any further.

If the driver has been downloaded directly from QDevil, we recommend placing both the driver itself and the Jupyter notebook in a folder separate from the QDoDeS installation so that it will not be overwritten when QCoDeS is updated. It will then be necessary to add this folder to the python path (for example `c:/users/me/python/mydrivers`):

In your Python file using the driver and in the included Jupyter notebook add the following lines in the top:

```
import sys
sys.path.append("c:/users/me/python/mydrivers/")
```

Then replace this line

```
import qcodes.instrument_drivers.QDevil.QDevil_QDAC as QDac
```

by

```
import QDevil_QDAC as QDac
```

3 Functionality

Each BNC output is regarded as an *instrument channel* in QCoDeS. Each channel has a number of parameters which can be “get” and most also “set”. A QDAC channel is referenced as qdac.ch##.<parameter>, where ## is the BNC number. The included examples illustrate how to use the driver.

All standard functionality of the QDac are available in this driver, this includes; setting and reading output voltages, reading currents, setting output ranges and current measurement ranges. Furthermore, the driver utilizes the 8 waveform generators to ramp voltages of up to 8 channels simultaneously. Assigning a finite “slope” to a channel protects the sample by setting voltages gradually by ramping with the defined slope. The 8 function generators are shared among all channels but only 8 may be changed simultaneously. A pulse can be output on one of the SYNC ports at the start of a ramp.

Note that when the output voltage range is changed by changing “mode”, the QCoDeS range check is also updated. QCoDeS throws an exception when trying to set a parameter value outside its valid range. A parameter’s valid range is given by the parameter’s “vals” attribute.

Overview of channel commands (accessed as qdac.ch##.<parameter>)

Channel Parameter	Meaning
.v()	Sets or reads the output voltage (Volts)
.i()	Reads the current flowing out (Amperes)
.mode()	Sets or reads both the output voltage range and current sensor range, see below for details*.
.slope()	Sets or reads the ramp rate of a channel in Volts/sec. Set to ‘Inf’ to release the generator occupied by the channel, and to stop ramping this channel.
.sync()	Defines on which SYNC output a pulse should occur when the channel is ramped.
.sync_delay()	Defines the delay for the sync pulse (seconds)
.sync_duration()	Defines the duration of the sync pulse (seconds)

***) mode**

The qdac.ch##.mode commands set or reads both the voltage output range and current sensor range, as these are inter-related. The mode parameter is an enum with the following values (see “how” in the Jupyter notebook):

Constant	Meaning
Mode.vhigh_ihigh	High voltage range, high current range.
Mode.vhigh_ilow	High voltage range, low current range.
Mode.vlow_ilow	Low voltage range, low current range.

Other commands (accessed as qdac.<command>)

Command	Meaning
.ramp_voltages()	Simultaneously ramps up to 8 channels from one set of voltages to another set of voltages .
.ramp_voltages_2D()	Function for smoothly ramping two channel groups simultaneously with one slow (x) and one fast (y) group (for example for charge stability diagrams). Function generators and triggers are assigned automatically. A total of 8 channels can be swept.
.temp_x_y()	Reads temperature sensor #y on board #x.
.cal()	Routes the specified channel to the CAL BNC connector.
.mode_force()	Default False, prohibiting the user from changing the voltage range by changing the mode parameter when the output voltage is non zero. When True the output voltage is not checked when the mode parameter is changed.
.channels[]	Used for addressing multiple channels at the same time, Index is 0-based.
.print_overview()	Prints a formatted overview of the state of all channels.
.print_slopes()	Prints a list of the slopes assigned to all channels.

.print_syncs()	Prints a list over which channels have been assigned which SYNC ports.
.reset()	Resets the QDAC to power on state. The reset command also flushes the communications buffer to recover from a QDAC power off/on event without the need for restarting the driver.

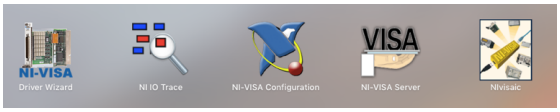
4 Getting started with the driver

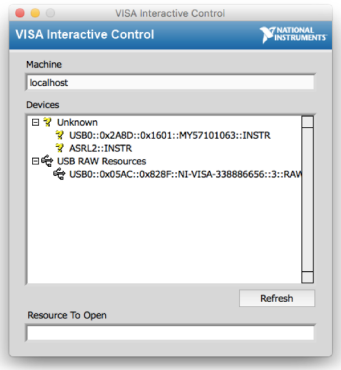
It is expected that the QCoDeS package is installed and functional.
Internally QCoDeS uses the National Instruments VISA software stack and the 'PyVISA' python package, it is expected that these are installed and functional.

The QDac driver is initialised by the following call:

```
qdac = QDac.QDac(name = 'qdac', address = 'ASRL2::INSTR', update_currents=False)
```

The QDac instrument address is easily found by running "NIvisaic" (NI VISA Interactive Control). The QDAC will be listed as "ASRL##::INSTR", where ## is the serial port number. In case you have multiple usb/serial port devices, you may need to unplug and re-plug their communication wires to find out which port belongs to which instrument.





When the driver initialises, it starts by reading the entire state of the QDAC. By default, updating the readings from the current sensors is omitted as it takes about 15 seconds for a 24-channel QDAC and 30 seconds for a 48-channel unit. If you should want the current sensors to be read at start-up, it can be enforced this by changing the `update_currents` parameter from "False" to "True":

```
qdac = QDac.QDac(name='qdac', address='ASRL2::INSTR', update_currents=True)
```

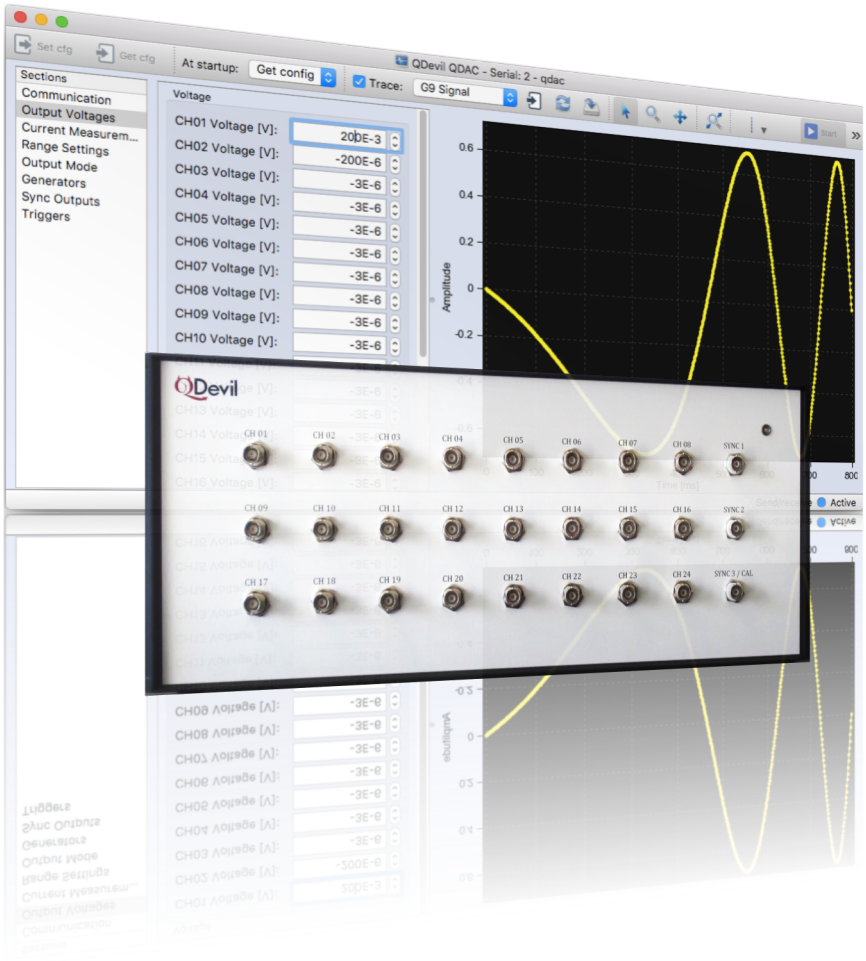
Now the QDAC can be operated, please see more examples in the "QCoDeS Example with QCodes examples with QDevil_QDac.ipynb" Jupyter notebook:

```
# Set voltage of a channel
output_voltage = 0.6
qdac.ch01.v(output_voltage)

# Set the slope of channel 01 to 1V/s and sweep a bit between values,
qdac.ch01.slope(1)
qdac.ch01.v(1.0)
time.sleep(2)
qdac.ch01.v(-1.0)

# Read and print the current measured for a channel
current_meas = qdac.ch01.i()
print("Measured current: {} uA".format(current_meas*1e6))
```

QDAC Labber Driver



Author: AK
Date: 09-05-2019

Manual version: 1.0b
Driver version: 1.01

1 Installing the driver

The driver consists of the following files which are zipped together:

QDevil QDAC.ini	The Labber driver definition file (default driver)
QDevil QDAC With SW Ramping.ini	Alternative Labber driver definition file enabling software sweeping*
QDevil_QDAC.py	The custom Python code part of the driver
qdac.py	Python interface to the QDAC, version 1.22
AWGcurve.txt	Example curve for the AWG

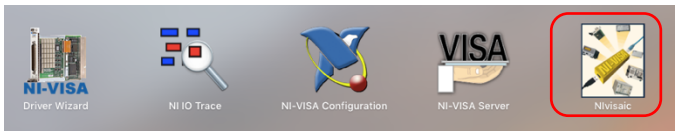
These files should be placed in a subfolder “QDevil QDAC” in the Labber driver directory in your user folder or in the driver folder in the Labber installation folder. Please see the Labber documentation for details. The driver has been tested with Labber version 1.6.3

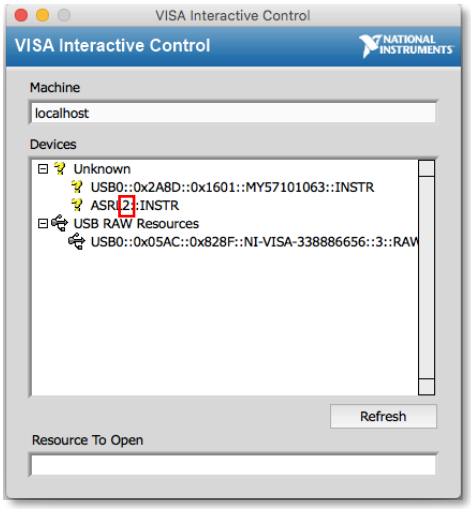
*)In order to minimize the number of controls in the driver user interface “QDevil QDAC.ini” driver does not enable software ramping. If software ramping between values in step sequences is desired, please use the “QDevil QDAC ramp.ini” driver instead – otherwise you may delete it.

2 NI-VISA – finding the device address

The QDAC Labber driver uses the NI VISA framework for communication using usb/serial port on the computer. So, it is necessary to install NI VISA on the computer running Labber. Please see the Labber documentation for details.

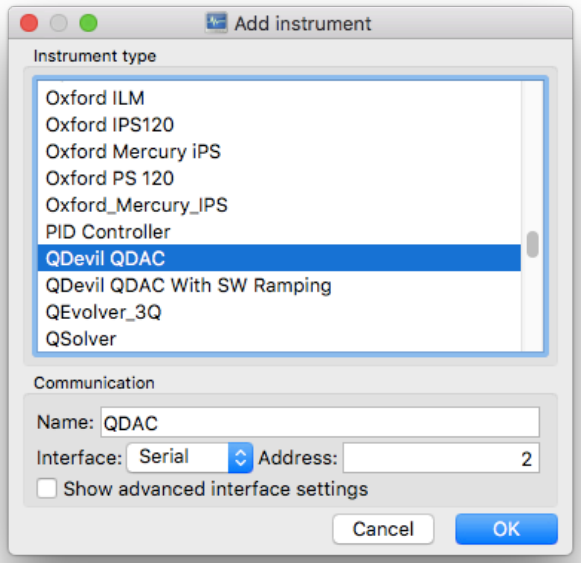
To find the port number of the QDAC, please run “NIvisaic” (NI VISA Interactive Control). The QDAC will be listed as “ASRL##::INSTR”, where ## is the port number. In case you have multiple usb/serial port devices, you may need to unplug and re-plug their communication wires to find out which is which.





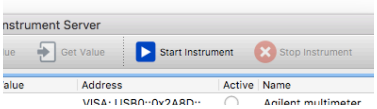
3 Add the QDAC to the Instrument Server

In the Labber Instrument Server, add the QDAC by clicking the “Add” button. The driver is named “QDevil QDAC” (the alternative driver is “QDevil QDAC With SW Ramping”). Select it in the list, enter a name, and enter the serial port number found in NI VISA in the Address field. Finally click OK.



4 Starting the QDAC

Adding the QDAC to the Labber Instrument Server allows browsing all the available options and functionality of the driver and generating a configuration file; but, it does not initiate actual communication with the instrument. Push the “Start” button in the Instrument Server dialog or in the Config dialog to connect to the QDAC.



“Starting” the QDAC takes about 15 seconds for a 24 channel QDAC and 30 seconds for a 48 channel QDAC, because the Labber driver will first read the complete state of the QDAC – except for the curve data for the AWG as this is not currently readable. Especially reading the current sensors takes time. You can see when Labber has completed reading the QDAC state by watching the LED on QDAC front panel: When its stops blinking red, the communication is completed.

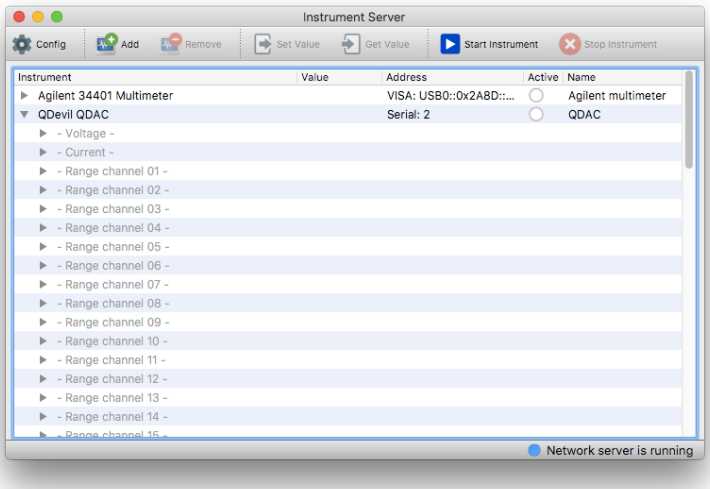
The reason for reading in the current state of the QDAC is to be able to recover from computer- or software crashes.

If you do not want the driver to read-in the hardware state at start-up, edit the QDevil QDAC.ini file and delete or comment out the “startup: Get config” line in the top section. Then remove the QDAC from the Instrument Server and add it again in order to apply the changes to the ini file.

5 Controlling the QDAC

The described version of the Labber driver provides access to most of the QDAC's. Functionality, not accessible is for example instrument calibration and soft synchronization. Please see the QDAC manual for a full description of the QDAC's functionality.

The QDAC can, as other Labber instruments, be controlled directly from the Instrument Server dialog using the "Get Value" and "Set Value" commands for each quantity. All QDAC quantities (settings, output voltages, and measured currents) are listed in various expandable groups in the Instrument Server dialog. All output voltages (24 or 48) are placed in one group and all measured currents in another. This makes it easy to overview them both in the Instrument Server dialog and in the Measurement dialog than if all options for each channel were placed together.



5.1 The Config dialog

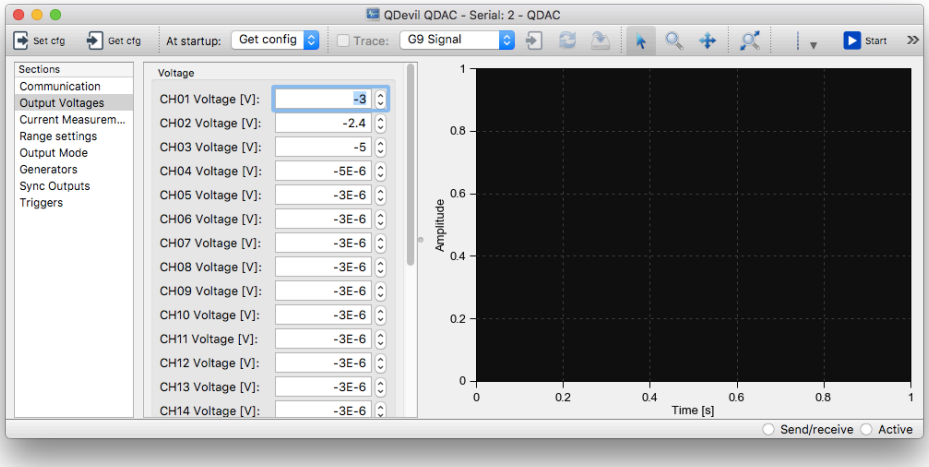
The Config dialog provides a convenient front panel for the QDAC. In here the settings and values have been grouped further into sections, making it easier to find controls for this instrument which has a much larger number of input/outputs than most other instruments have. Please consult the QDAC manual for functionality details.

In the left side of the Config dialog you find the sections into which the various controls are grouped. When one of these is clicked the corresponding controls appear in the middle.

In the right side there is a chart area. This is used for displaying any curve uploaded to the QDAC's arbitrary waveform generator and appears empty until a curve has been loaded from disk.

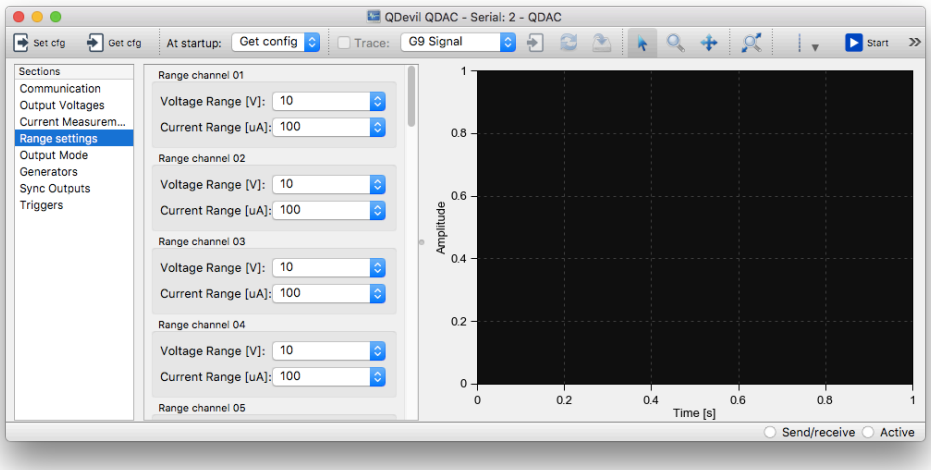
Example: Setting a voltage

To set a voltage on one of the QDAC outputs, simply go to the “Output Voltages” section and enter the new voltage in the field for the desired channel.



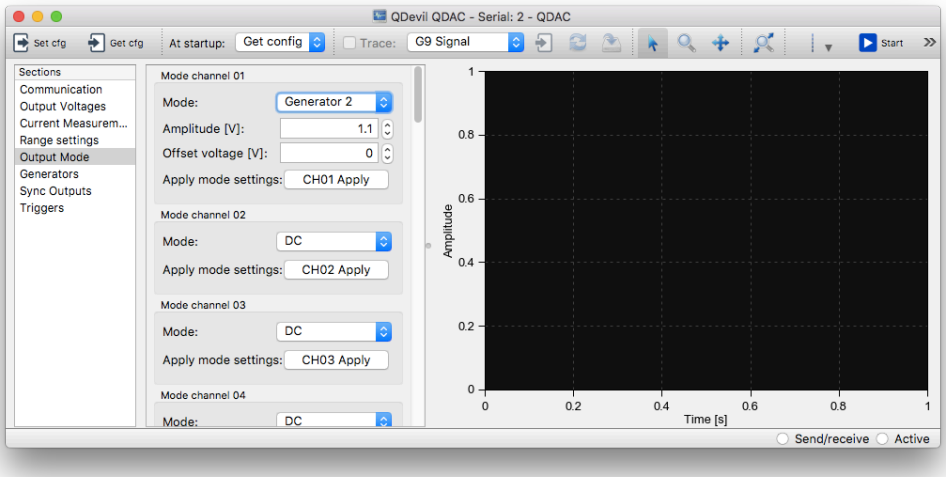
To change the output voltage range (or the current measurement range) go to the “Range settings” section.

Note that the QDAC cannot be in the low voltage range and high current range at the same time.



Example: Outputting a waveform

To output a waveform, first go to the “Output mode” section. In the Mode pull down menu select which generator should be connected to this channel. Then set the amplitude and offset for the generator (see QDAC manual). Finally push the “CH## Apply” button to transfer the “Mode” settings to the QDAC. The reason why the Mode setting is not applied immediately when it is changed is the following: IF a generator is already running then switching a channel’s Mode to output that generator could result in applying an undesired high amplitude or offset to the output. So therefore, one needs to push “Apply”, as this gives you a chance to first adjust offset and amplitude.

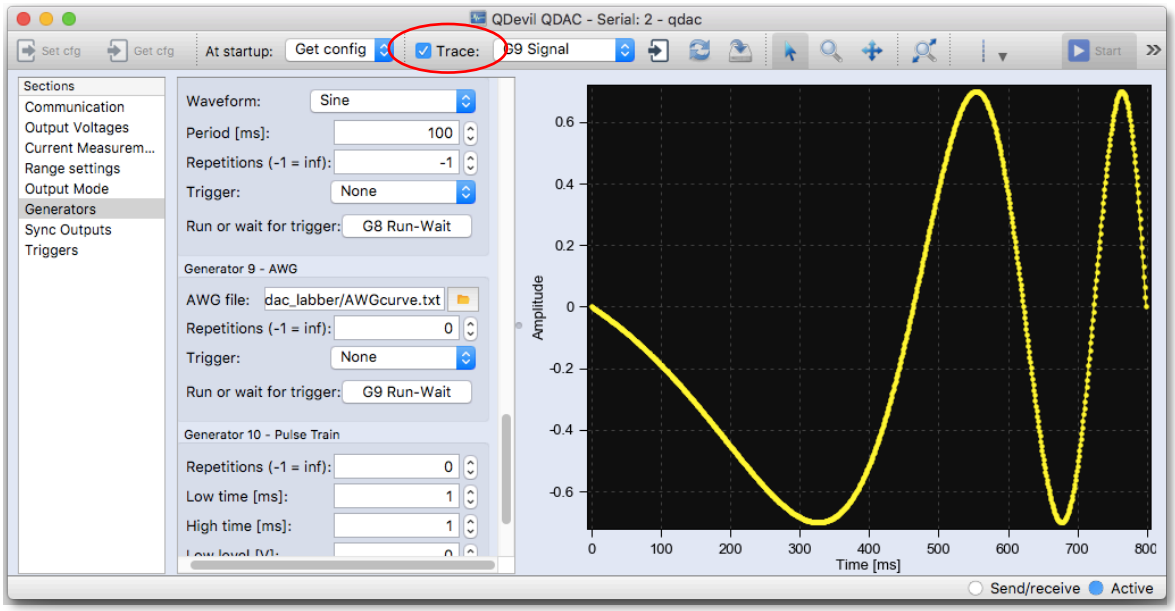


Then got to the “Generators” section and configure it. To program and start the generator using the displayed leave the “Trigger” field to “None” and push the “Run-Wait” button. If you want to start several generators at the simultaneously select the same trigger for those generators. Then got to the “Triggers” section and push the relevant trigger “Fire” button.

Note: Setting repetitions to zero and pushing the “Run-Wait” button will stop a running generator. If “Trigger” is different from “None” the generator will also stop when the “Run-Wait” button is pushed.

Uploading a curve data to the AWG generator

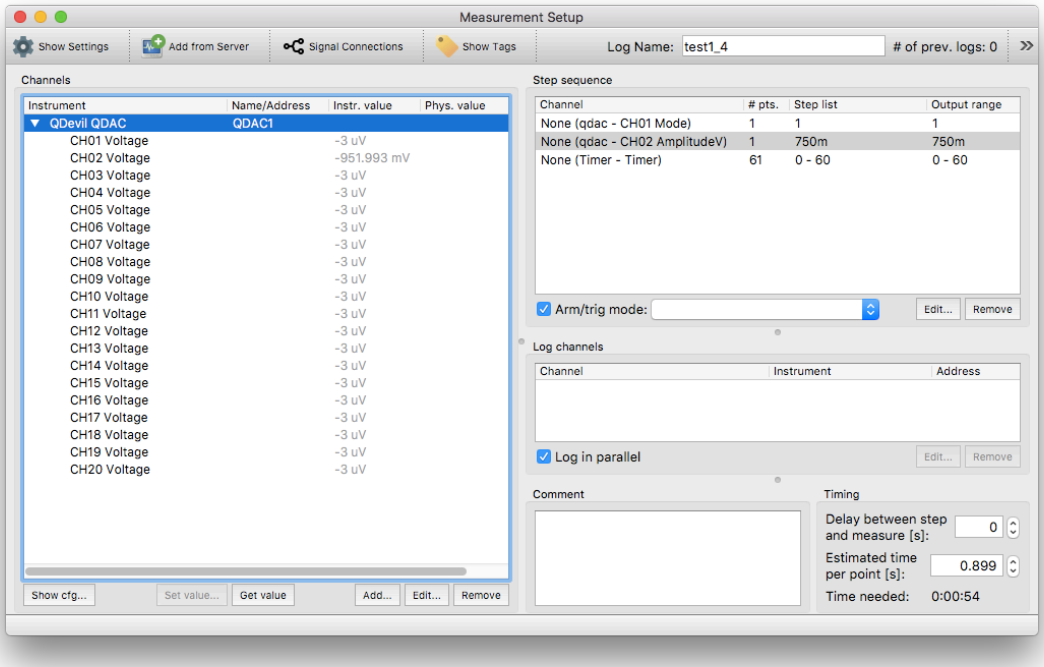
To upload an array of curve data to the arbitrary waveform generator, click the “AWG file” icon and browse to a file which has one column of numerical values defining the samples for the AWG. Select and open the file. The check the “Trace” check box in the menu bar in the dialog. This will actually load and display the curve. Each sample will be 1 ms (one sample clock) long .

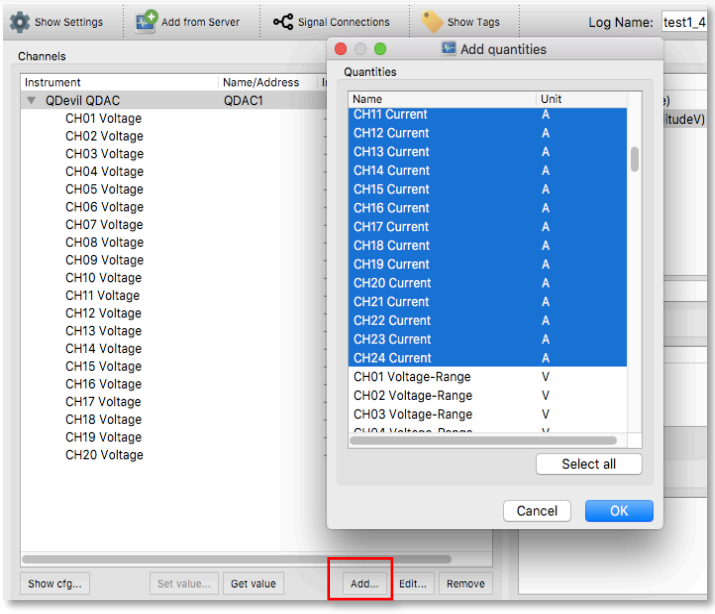


Now set the number of repetitions, and optionally a trigger, and click the “Run-Wait” button. This will upload the curve to the QDAC and start the AWG – or arm it if a trigger was chosen.

Measurement dialog – adding more channels

In Labber version 1.6.3 and earlier the Measurement dialog will only show up to 20 quantities per instrument as default. This means that initially you will only see the voltage setting of channels 1-20. To enable all channels and also for example the current sensors, select the QDAC in the Measurement dialog, and click the “Add” button and add more channels.

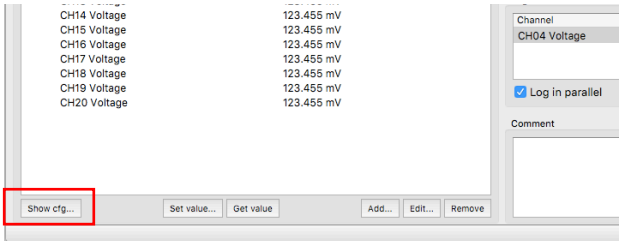


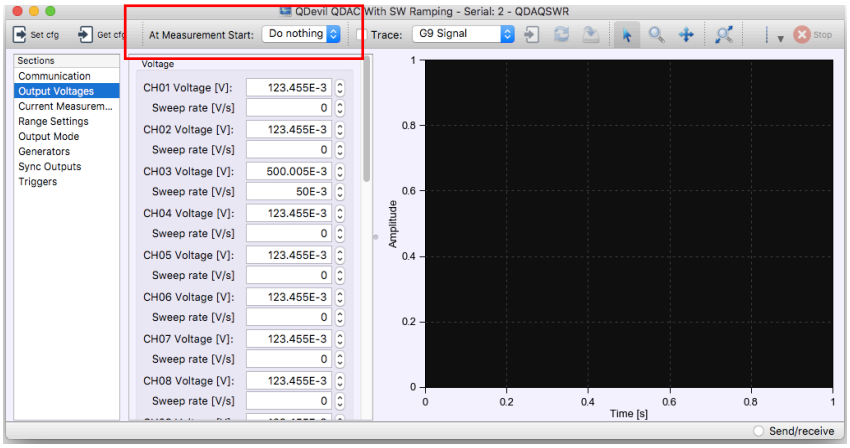


Note that push buttons cannot be added to a step sequence. Therefore, function generators cannot be programmed and started, and the output mode of channels cannot be changed within a step sequence.

When “Starting” a measurement the instrument will initialize. For the QDAC it means that its current state will be read. This can take some time. To avoid go to the Config dialog by pushing “Show cfg…” in the Measurement dialog, and then set the “Set Measurement Start” option to “Do nothing”. This is not recommended for general usage. But when testing sequences, it may become annoying having to wait for

the Labber driver to read the QDAC state. Alternatively, the “Get config” behavior may be disabled entirely by changing the “startup” option in the ini file as mentioned earlier.





The Config dialog when started from the Measurement dialog. Note that in this screen shot there are sweep rates listed for each output voltage. This is because the “QDevil QDAC With SW Ramping.ini” driver was chosen instead of the default driver.