

# QDAC-II Operation Manual



24-channel DC voltage source with:

- Integrated waveform generators
- DC current sensors on each channel
- Up to 25-bit resolution in DC mode

**Please start your QDAC-II experience by first installing the latest firmware**

Get the latest QDAC-II pdf manual, firmware update and other information  
by following this link or the QR code: <https://qm.quantum-machines.co/87kjeif6>



## Known bugs and important limitations in firmware version 13-1.57

Below is a list of known bugs and of planned functionality which have not been implemented yet at the day of publishing this manual. QDevil continues to work hard on fixing any reported bugs and on completing the firmware and will publish new firmware versions frequently over the next period of time. We appreciate any reports of additional bugs which may be found. Please send them to [sales@qdevil.com](mailto:sales@qdevil.com). Some minor known bugs are not reported below.

### Communication

- To make the instrument acquire an IP address via DHCP, the ethernet cable should be plugged in before powering up the unit, or it should be restarted after plugging in the cable (SYSTem:REStart).
- In addition to executing a SYSTem:COMMunicate:LAN:UPDate when LAN settings are changed a SYSTem:REStart is also required in order for changes to take effect.
- When uploading vast amount of binary data over LAN, for example multiple large (4-6 mio points) arbitrary waveform traces), the instrument may occasionally report an error and the upload will fail. It happens if there are some instabilities in the local network, but other times this is not the cause. So always check for errors after uploading a trace, and in case of errors, re-upload the trace.
- Channel list notation only works for the last command in a compound command.

### Functionality

- Please do not change the voltage range (RANGe) at non-zero output voltages, and always immediately explicitly set the DC voltage to zero (again) after changing the range, as some mechanisms required when changing ranges are not fully automatic yet.
- Triggering of DC voltages (SOUR:VOLT:TRIG level) in FILTER = DC (resolution-enhanced) mode is so far not working correctly.
- Please avoid setting SOUR:VOLT:TRIG to a new value immediately before switching in and out of DC mode while resolution enhancement (RENH) is enabled (which it is by default), as the TRIG value erroneously will be applied when switching the filter in and out of DC – which could cause sample damage.
- Current limitation is not implemented so far. Please do not try to source more current than around 10mA for each channel.
- Trigger outputs are appearing 1-4  $\mu$ s *before* voltage outputs triggered by the same sources, e.g. waveform markers or trigger inputs, due to delays in the analogue circuits. To compensate (to best sub- $\mu$ s level) please add a trigger output delay.
- In the current version of the firmware the LAST command does not work as designed, as it will report the value last set by the VOLT:TRIGger command even if the generator has not been triggered.
- Setting the voltage output of channel using the raw SOURce:DAC command will not set a 25 bit value in DC mode (where 25 bit resolution enhancement is active). A fractional DAC value will be rounded to its nearest integer.
- Queried voltages are slightly off (by less than 20 $\mu$ V; or 4 $\mu$ V in the 2V range) due to internal rounding errors. The same is true for queried raw DAC values.

- NCLeft is reporting “-1” for generators which have been ABORted if they have COUNT set to Inf. But they *should* return”0”.
- Continuous triggering (INIT:CONT = ON) of the DC generator in DC filter mode (when resolution enhancement (RENH) is on, does not work. One time triggering following an INIT command works fine. Bug appeared in version 13-1.54.

## Table of contents

<b>QDAC-II OPERATION MANUAL .....</b>	<b>1</b>
<b>1 SAFETY AND REGULATORY INFORMATION .....</b>	<b>7</b>
<b>2 MANUAL OVERVIEW .....</b>	<b>8</b>
<b>3 INTRODUCTION TO THE QDAC-II .....</b>	<b>9</b>
3.1 BASIC DESCRIPTION .....	9
3.2 COMMUNICATION .....	14
3.3 FIRMWARE UPDATE .....	16
<b>4 GETTING STARTED .....</b>	<b>17</b>
4.1 POWERING UP QDAC-II USING THE QDEVIL POWER SUPPLY .....	17
4.2 CONNECTING THE QDAC-II TO YOUR EXPERIMENT .....	18
4.3 COMMUNICATING WITH THE INSTRUMENT .....	19
4.3.1 ETHERNET COMMUNICATION .....	19
4.3.2 USB/SERIAL COMMUNICATION .....	20
4.4 PYTHON .....	24
4.4.1 QCoDES (PYTHON) .....	25
4.4.2 LABBER (PYTHON) .....	25
<b>5 FEATURES AND FUNCTIONALITY .....</b>	<b>26</b>
5.1 CHANNEL CONFIGURATION .....	26
5.1.1 VOLTAGE RANGE SELECTION .....	26
5.1.2 LOW PASS FILTERS, OUTPUT IMPEDANCE .....	27
5.1.3 SLEW RATE .....	28
5.1.4 CURRENT RANGE AND CURRENT LIMITATIONS .....	29
5.2 VOLTAGE GENERATION .....	30
5.2.1 DC SIGNAL GENERATION .....	30
5.2.2 FIXED WAVEFORM GENERATORS .....	36
5.2.3 ARBITRARY WAVEFORM GENERATOR (AWG) .....	43
5.3 CURRENT SENSING .....	45
5.3.1 CURRENT MEASUREMENT .....	45
5.4 TRIGGER SYSTEM .....	49
5.4.1 STARTING (TRIGGERING) AND STOPPING VOLTAGE GENERATORS .....	50
5.4.2 TRIGGERING OF CURRENT SENSORS .....	50
5.4.3 TRIGGER OUTPUTS AND INPUTS .....	50
5.4.4 INTERNAL TRIGGERS AND MARKERS .....	51
5.5 SYNCHRONIZATION OF MULTIPLE QDAC-II UNITS .....	54
<b>6 OPERATION .....</b>	<b>57</b>
6.1 SCPI COMMAND GROUPS .....	57
6.2 SCPI COMMAND SYNTAX .....	58
6.3 COMMAND SYNCHRONIZATION .....	60

6.3.1	SEQUENTIAL VERSUS OVERLAPPED COMMANDS .....	60
6.3.2	FUTURE SECTION ON SOFTWARE SYNCHRONIZATION (PLANNED FEATURES) .....	60
6.3.3	HARDWARE SYNCHRONIZATION.....	60
6.4	STATUS AND EVENTS .....	61
6.4.1	FUTURE SECTION: STATUS:OPERATION .....	62
6.4.2	FUTURE SECTION: STATUS:QUESTIONABLE.....	62
6.5	ERRORS AND EVENTS REPORTING .....	63
6.5.1	BUZZER AND LED.....	63
<b>7</b>	<b>COMMAND REFERENCE .....</b>	<b>64</b>
7.1	IEEE COMMON COMMANDS.....	64
7.2	FUTURE SECTION: INTERFACE COMMANDS.....	66
7.3	VOLTAGE GENERATORS.....	67
7.3.1	OUTPUT MODES.....	67
7.3.2	DC GENERATOR.....	68
7.3.3	SINE WAVE GENERATOR.....	80
7.3.4	SQUARE WAVE GENERATOR.....	85
7.3.5	TRIANGLE GENERATOR .....	91
7.3.6	AWG GENERATOR .....	96
7.3.7	TRACE (AWG) CURVE MANAGEMENT .....	99
7.3.8	TRIGGERING OF VOLTAGE GENERATORS .....	101
7.4	COMMANDS FOR CONTROLLING TRIGGERS .....	104
7.4.1	INTERNAL TRIGGERS AND MARKERS .....	104
7.4.2	TRIGGER OUTPUT CONFIGURATION .....	108
7.5	CURRENT SENSOR COMMANDS .....	111
7.6	SYSTEM COMMANDS .....	120
7.6.1	COMMUNICATION SETUP COMMANDS .....	120
7.6.2	ERROR SYSTEM .....	124
7.6.3	OTHER COMMANDS.....	125
7.7	DIAGNOSTIC COMMANDS .....	129
7.8	FUTURE SECTION: REPORTING OF OPERATION STATUS.....	133
7.9	FUTURE SECTION: QUESTIONABLE DATA (CURRENT LIMIT) .....	133
<b>8</b>	<b>FUTURE SECTION: LIST OF ERROR MESSAGES.....</b>	<b>133</b>
<b>9</b>	<b>THE IEEE 488.2 BINARY BLOCK FORMAT .....</b>	<b>134</b>
<b>10</b>	<b>SPECIFICATIONS AND PERFORMANCE .....</b>	<b>135</b>
10.1	NOISE AND DRIFT .....	136
10.2	NON-LINEAR OUTPUT CHARACTERISTICS .....	138
10.2.1	LOW CURRENT MODE .....	138
10.2.2	HIGH CURRENT MODE .....	139
10.3	FILTER SWITCH TRANSIENTS .....	140
10.4	CURRENT MEASUREMENT SIGNAL TO NOISE RATIO.....	141










<b>11</b>	<b>BIBLIOGRAPHY.....</b>	<b>142</b>
<b>APPENDIX A</b>	<b>USING LABORATORY POWER SUPPLIES.....</b>	<b>143</b>
	APPENDIX A1. PROCEDURE FOR CONFIGURING A QL355TP POWER SUPPLY.....	143
	APPENDIX A2. PROCEDURE FOR TURNING ON THE QDAC-II USING A QL355TP.....	144
<b>APPENDIX B</b>	<b>OPEN-SOURCE LICENSES.....</b>	<b>145</b>

## 1 Safety and regulatory information

This device has been tested and has been supplied in a safe condition. This manual contains some information and warnings which must be followed by the user to ensure safe operation and to retain the instrument in a safe condition.

This device has been designed for indoor use in the temperature range 15°C to 30°C, **preferably 20-25°C**, 20% - 80% RH (non-condensing). Do not operate while condensation is present. When bringing the device from a cold environment to a warm environment, please allow the unit to thermalize (2-4 hours) before taking it out of its shipping box and attempting to power it on.

Use of this instrument in a manner not specified by these instructions may impair the safety protection provided. Do not operate the instrument outside its rated supply voltages or environmental range.

	<p>Keep this device away from children and unauthorized users.</p>
	<p><b>Indoor use only.</b> Keep this device away from rain, moisture, splashing and dripping liquids. Never put objects filled with liquids on top of or close to the device.</p>
	<p><b>DO NOT</b> disassemble or open the cover without first getting advice from QDevil.</p>
	<p><b>Caution:</b> Device heats up during use. Make sure that the ventilation openings at the bottom and rear plates of the instrument are clear at all times. Place the device on a flat, heat resistant surface, do not place the device on carpets, fabrics etc. To avoid over heating allow some spacing to other heat generating devices (e.g. other QDAC-IIs).</p>
	<p><b>Please</b> only connect the instrument to the included power supply or other approved power sources using the cable(s) delivered with the instrument.</p>
	<p>Keep this device away from dust and extreme temperatures.</p>
	<p>Protect this device from shocks and abuse. Avoid brute force when operating the device.</p>
	<p>Do not use the device when damage to housing or cables is noticed. Do not attempt to service the device yourself but contact QDevil.</p>
	<p>Not for general waste, recycle as electronic waste</p>

## 2 Manual overview

Please see first page for known issues and limitations of the current version (13-1.57) of the firmware and this manual.

### 1 Safety and regulatory information

Important precautions and requirements to the environment in which this instrument is used.

### 2 Manual overview

This section.

### 3 Introduction to the QDAC-II

To get to know the instrument it is highly recommended starting by reading this. A feature overview is provided mentioning the unique features of the QDAC-II as well some of the limitations of the instrument.

### 4 Getting started

This section provides the information required to get the instrument hooked up so that you can try out some of the examples in the “5 Features and functionality” section.

### 5 Features and functionality

Describes the essential functionality and the associated SCPI commands. Examples are also provided. It does not give a full description of the command syntax. For that please see “7 Command Reference”.

### 6 Operation

Introduces the SCPI protocol, command sequencing and status and errors.

### 7 Command Reference

Is a complete reference for all commands.

### 8 Future section: List of error messages

This section will in the future contain a detailed list of error messages

### 9 The IEEE 488.2 binary block format

Contains a description of the binary block format used for transferring AWG traces and long DC LISTs.

### 10 Specifications

Contains a specification overview and data sheet curves.

### 11 Bibliography

Is a list of relevant references.



### 3 Introduction to the QDAC-II

The QDAC-II is a 24 channel 20-bit voltage source with a 1 million samples per second output rate, where each BNC output has five voltage generators working in parallel, plus a current monitor:

- One DC voltage source with sweep and list capabilities.
- One sine wave generator.
- One triangle / saw tooth generator.
- One square wave / pulse generator.
- One arbitrary wave form generator.
- Current sensing at 3 kHz speed (with  $\approx 1.5$  ms integration time).

Each channel has two voltage ranges and two current ranges and three low pass filter options. In its low bandwidth “DC” mode, the resolution is enhanced to 25 bits (see section 10) when setting fixed DC levels (in DC:FIXed mode).

The bandwidth of the waveform generators is limited by the sample rate and the low pass filters (see below).

An advanced triggering system with internal as well as external routing makes it possible to generate advanced signals and sequencies with accurate timing and external synchronization.

The number of channels can be enhanced by synchronizing multiple QDAC-II units so that voltage changes and waveforms are coordinated with sub-microsecond precision over several units.

#### 3.1 Basic description

##### Front panel

On the front panel the 24 voltage outputs are located. They are standard BNC connectors with their shield connected to the common chassis ground. The output series resistance is 50 ohms.



Figure 1. Front plate of the QDAC-II. Here, all voltage outputs and three isolated trigger outputs are found.

In addition, three trigger outputs (no. 1-3) are in the right most column. These have their shield and signal galvanically isolated from the rest of the QDAC, so that ground loops are avoided when connecting these outputs to other instruments. The voltage level is 4.8V with an output impedance of approximately  $50 \Omega$  ( $\pm 20 \Omega$ ) and will provide around 3.3 V in 50 load. This means that it is safe to connect Trig. Out 1-3 to a terminated  $50 \Omega$ , 3.3V input.

## Rear panel

Most important on the rear panel are the Power connector and the USB and Ethernet connectors. The power is supplied by the QDevil QPSU which is a linear power supply with galvanic isolation to mains ground. Both the LAN and USB connectors feature galvanic isolation for both ground and signals, so that there is no risk of creating ground loops that way, or of introducing noise to the QDAC galvanically.

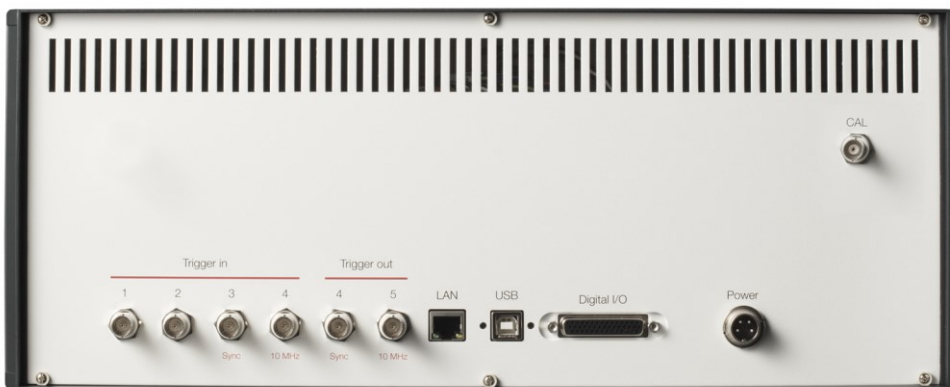


Figure 2. Rear plate of the QDAC-II. On the rear panel the power, USB, and ethernet connections are found plus additional trigger in and out connectors, including a 44 pin sub-D connector for additional future functionality.

In addition to the trigger outputs on the front panel, there are two more trigger outputs on the rear panel. Note that these two are not galvanically isolated. These outputs have a 3.3V level in a high impedance. The output impedance is not specified, but they will give up to around 2.1V in a  $50 \Omega$  load suggesting an effective output impedance of around  $30 \Omega$ . The same two outputs are used when the unit is the master in a stack of synchronized QDAC-II units.

The four trigger *inputs* (no. 1-4) are all galvanically isolated and used for triggering the voltage generators or current sensors in the QDAC-II from external devices. No. 3 and 4 are also used as inputs for clock and synchronization when the unit is a slave in a stack of synchronized units. The input impedance is  $10 \text{ k}\Omega$ , the maximum voltage is 3.8 V, and the minimum voltage  $\pm 0.5 \text{ V}$ . The inputs react on positive going edges. The minimum guaranteed trigger level is +2.2 V and should preferably be +2.5 V.

The Digital I/O connector (44 pin high density D-sub) is a multi-purpose connector for future expansion. It includes 3.3V and 5V power lines for external electronics.

## Architecture

Very simplified the QDAC-II can be considered to be made up by five functional blocks:

- Channel outputs
- Waveform generators
- Trigger inputs
- Trigger outputs
- Trigger system

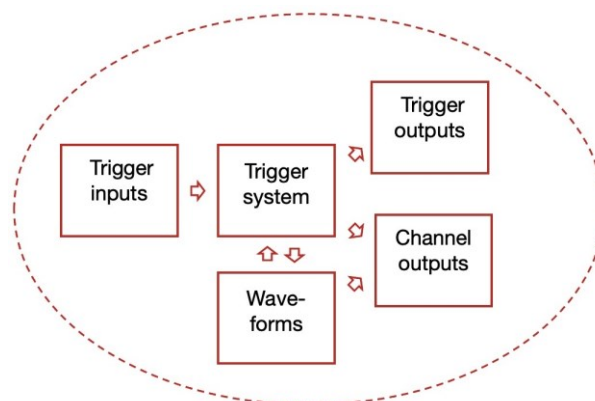


Figure 3. Functional architecture of QDAC-II

## Channel outputs

### Overview

Each of the 24 channels consists of a high precision DAC (digital to analog converter) working in two selectable voltage ranges ( $\pm 10$  V and  $\pm 2$  V). The standard resolution is 20 bits.

The signal chain has two stages of lowpass filtering, one of them appearing at the final output. The two stages are hard-linked so that they switch simultaneously. The combined filter has three selectable cut-off frequencies, HIGH  $\approx 300$  kHz, medium  $\approx 10$  kHz, DC  $\approx 10$  Hz.

In the “DC” filter mode the DAC resolution is enhanced to 25 bits when setting fixed DC levels (in DC:FIXed mode).

In between the two lowpass filter stages is a current sensor with two ranges, a coarse range (HIGH) used for currents up to  $10\text{mA}^1$ , and a fine range (LOW) for very small currents (up to  $200\text{nA}$ ), useful for detecting leaks. In the fine range there are some limitations to the bandwidth of the voltage output, which are a little bit subtle please see section 5.1.4.



Figure 4. Key function blocks of each channel on QDAC-II: D/A converter with two ranges, low-pass filtering with three cut-offs, and current sensing with two ranges.

<sup>1</sup>  $10\text{mA}$  nominally as default, but can be tweaked up to  $20\text{ mA}$  on a few channels.

## Output impedance

The low pass (LP) filter closest to the BNC is an RC filter with a 50  $\Omega$  output resistor. The capacitance is determined by which low pass filter is invoked and the capacitance of the load, see more in section 5.1.2.

## Low noise and drift

Detailed drift and noise data are provided in section 10.1. After switching on the instrument please allow a warm-up of 3 hours, and preferably 24 hours to achieve temperature stability. After that the output is stable to within about  $\pm 2 \mu\text{V}$  at low voltages and currents, if the ambient temperature is kept stable within  $\pm 0.5 \text{ }^\circ\text{C}$ .

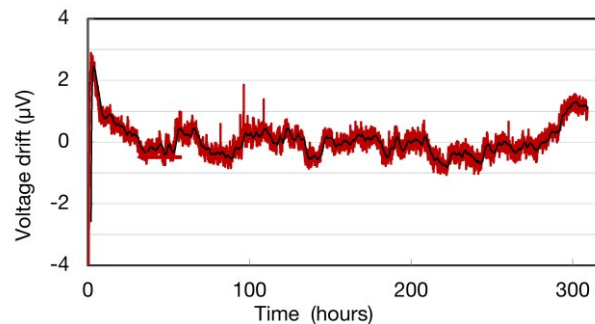


Figure 5. Change in output voltage recorded from 2-3 minutes after power on at 1V output in DC mode and 2V range.

## Low transients, device protection

All DAC chips working by the resistor ladder principle (R2R) exhibits a transient – *glitch* – at certain transitions (when major carry in the binary code take place). In QDAC-II this effect has been minimized so that the maximum glitch transient amplitude is around 1.2 mV, and is only observable in the HIGH lowpass filter mode. In the MEDium and DC filter modes the glitch transients are not observable.

The lowpass filters are analogue RC filters which inherently will cause a transient to be observed at the output when switched in and out. However, a key feature of the QDAC-II is that a special circuit (patent pending) allows switching lowpass filter range with a minimal transient appearing at the output. The maximum transient to appear, when switching filters at a DC output voltage of  $\pm 9.9\text{V}$ , has been measured to maximum  $\pm 1.5 \text{ mV}$  in a 350 MHz bandwidth, see details in section 10.3.

To assist the user in not accidentally applying steep voltage steps to the device under test all channels can be individually slew rate limited – at the generator level. The analogue lowpass filters will of course have a comparable function.

## Current sensing resolution

The current sensor signal to noise ratio (resolution) allows detection of currents steps down to 50pA or better, and often better, (at 1PLC integration time) in LOW current mode and down to 10  $\mu\text{A}$  or better in HIGH current mode. Note that this on the time scale of seconds or minutes. At longer time scales 1/f noise and drift plays a role making the long-term signal to noise ratio a bit worse

Note that the signal to noise ratio is, perhaps a bit surprising, higher in FILter=HIGH mode than in FILTer=MEDium or DC.

In addition, in both FILTER=DC and FILTER= MEDIUM modes the measured current will have a long settling time due to the slow relaxation of trapped charges in the output capacitors. This is possible to observe in the LOW current mode, especially following large voltage steps (see warning below).

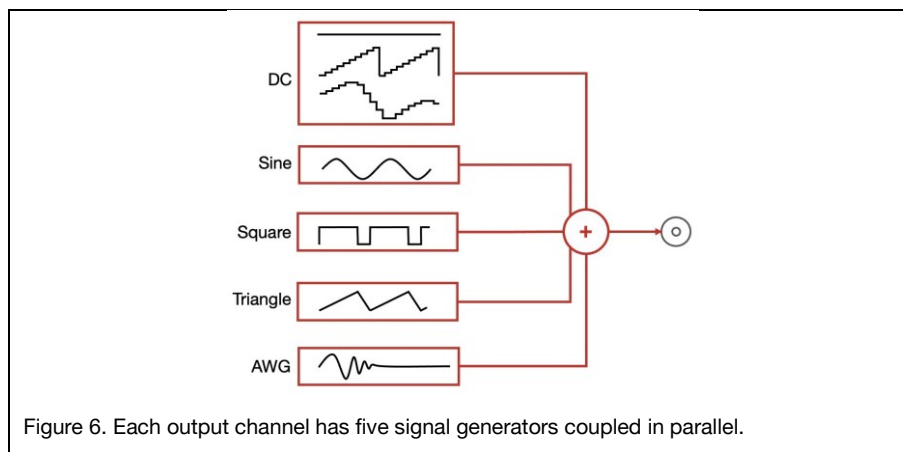
So it is a tradeoff: with low voltage noise follows high current noise and very long response times. Please see section 10.3 in “10 Specifications” for details and examples.

The accuracy is limited by the non-linearities described in section 5.1.4, but can be partly accounted for.

Note, that when FILTER=DC the time constant of the current sensor is of the order of minutes. Hence, it can not be recommended using current sensing in DC mode for other than very slow monitoring of the current at steady voltages. For getting the most precise readings FILTER=HIGH mode should be used for measuring current.

## Waveform generators

Each output channel is driven by five signal generators coupled in parallel as illustrated in Figure 6.



Each channel has its individual set of generators, so that all channels work independently from each other. In addition, their operation can be synchronized using the *trigger system*.

The DC generator can produce a DC signal, a sweep of equally spaced DC values or a list of arbitrary chosen DC values.

The Sine, Square and Triangle generators output waveforms according to their name with the possibility of adding a compensating DC offset. The Square signal generator is also useful as a pulse generator and has special settings for easily setting pulse width etc.

The arbitrary waveform generator gets its waveform data (traces) from a shared memory pool, meaning that multiple output channels can share the same waveform, but output it with individual amplitude and dc-offset for each channel.

Naturally the actual voltage output will depend on which low-pass filter cut-off has been chosen. In addition, the chosen maximum slew-rate will also modify the output.

## 3.2 Communication

The QDAC-II is controlled through its USB/serial interface or via ethernet – or both.

The USB/serial port speed is 1 Mbits/sec. It is recommended to use ethernet port, especially for trace uploads. For first time use (setting up LAN options etc) and for updating the firmware it is however necessary to use the USB/serial port.

The ethernet port has giga bit speed. For transferring traces for the AWGs, ethernet should preferably be used. It is a single TCP/IP socket connection and do as such not support IEEE488.2 status byte polling and service request generation. The instrument allows several clients to control the instrument. It is not recommended for multiple users to use the instrument simultaneously, though this is possible, as global commands (such as \*RST or ABORT) may disturb other users unintendedly – just think of what a \*RST will do. The multiuser feature is more intended for being able to share channels in multiple setups without having to disconnect and move the instrument around.

To communicate with the instrument, make sure to send command sequences one line at a time ending with a <LF> (Line Feed) termination character, hexadecimal code 'x0A'.

### USB/serial setup

The USB connector located on the rear panel of the instrument. The communication works through a virtual serial port over the USB interface, as there is a USB-to-serial adapter inside the instrument (opto-coupled). Since the USB connection galvanically isolated from the circuits inside the instrument opto-coupled, no transients in the output voltages will occur when the USB cable is plugged in or unplugged.

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 in the Getting started section, where the appropriate communication port settings are listed (section 4.3.2).

### LAN setup

From factory the LAN interface is set up to acquire an IP address, gateway and subnet mask using DHCP (Dynamic Host Configuration Protocol). The instrument will present itself with its hostname which by default is the same as its serial number.

It is also possible to manually specify set the LAN options by setting DHCP to OFF.

After making changes to LAN settings, the :LAN:UPDate command must be sent to the instrument in order for the modified settings to take effect and to store the settings in non-volatile memory.

Command	Description
SYSTem:COMMunicate:LAN:DHCP	Determines whether automatic LAN configuration is on or off (default ON)
SYSTem:COMMunicate:LAN:IPADdress	Sets or queries the instrument's IP address.
SYSTem:COMMunicate:LAN:HOSTname	Sets or queries the instrument's LAN name
SYSTem:COMMunicate:LAN:GATeway	Sets or queries the IP address of the gateway (router).
SYSTem:COMMunicate:LAN:SMASK	Sets or queries the sub-net mask.
SYSTem:COMMunicate:LAN:MAC?	Queries MAC address of the instrument.
SYSTem:COMMunicate:LAN:UPDate	Stores the LAN settings and restarts the LAN interface.

Note that with the current firmware version (13-1.57), to make the instrument acquire an IP address via DHCP, the ethernet cable should be plugged in before powering up the unit, or it should be restarted after plugging in the cable (SYSTem:REStart).

## SCPI Commands

The command set of the QDAC-II adheres to the SCPI standard with a few deviations.

One deviation is that the QDAC-II protocol does not support explicit units. Instead, all values are assumed to be in SI units (volts, amperes, seconds, hertz, etc.). This makes it simpler to write drivers. Some functionalities such as command synchronization is, however, not available in the first versions of the firmware.

### 3.3 Firmware update

Firmware updates are distributed as executables for multiple platforms. To perform the firmware update the instrument must be connected to the host computer via the USB/serial port. Please disconnect any program which may be connected to the device over USB/Serial before updating.

On MAC-OS and Linux systems the updater file must have its attribute changed to “executable” by using the “chmod +x” command. In addition, on MAC-OS it might be required to grant the updater file access in the Mac settings under *Privacy & Security* -> *Security* -> *Allow accessories to connect*.

The firmware update program will automatically identify the instrument and start updating. It lasts a couple of minutes. Please avoid interrupting the instrument and computer during the update.

Before starting the firmware updater program, make sure that you have disconnected any USB/Serial interface on your computer, which might be connected to the QDAC-II, for example a terminal program or Python code.

#### Example – executing the firmware installer on MAC-OS

```
> chmod +x qdac2-fw-update-macos
> ./qdac2-fw-update-macos

Updating QDAC-II firmware...
Deploying firmware version 4-0.9.20
14:25:15: Using serial device /dev/cu.usbserial-14240
14:25:15: Rebooting device...
14:25:23: Uboot version 1
14:25:24: Transferring APU0... (ETA 14:25:28)
14:25:39: Erasing Flash...
14:25:41: Writing Flash...
14:25:42: Transferring APU1... (ETA 14:25:46)
14:25:56: Erasing Flash...
14:25:58: Writing Flash...
14:25:59: Transferring PL... (ETA 14:28:11)
14:28:11: Erasing Flash...
14:28:21: Writing Flash...
14:28:33: Number of updates: 13
14:28:36: Rebooting device
(Press Enter to close)

Update finished
>
```

Please remember to push <enter> when finished, if prompted. Otherwise, a message will appear in your terminal/program when connecting over the USB/serial interface afterwards.

#### Web address for firmware updates

Updates, when available, can be downloaded from this web site: <https://qm.quantum-machines.co/87kjeif6>

#### Use of open-source libraries

The QDAC-II firmware incorporates open-source code, see Appendix B.



## 4 Getting started

As the first thing when receiving a new QDAC-II and after powering up, please check for new firmware updates and install the latest firmware (see Firmware update in section 3).

To avoid ground loops or other spurious circulating currents, it is highly recommended to use the enclosed rack-mount isolators when mounting the QDAC-II in a metallic rack. Avoid making the QDAC-II cabinet touch other instruments in the rack.

Please pay attention when rack-mounting the instrument using the included isolators, as the isolation on the 6 mm screws may move when pressed through the plastic “ears”. Please test the isolation before connecting the QDAC-II to anything else.

### 4.1 Powering up QDAC-II using the QDevil Power Supply

It is recommended to install the instrument in a normal laboratory environment (i.e. at non-extreme temperatures and humidity) with a stable temperature in the range of 18-25 °C. In order to achieve minimal noise (e.g. 50/60 Hz), the instrument 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 instrument as the cable permits and preferably not on the same vertical axis. 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.

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 -II instrument 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-II, 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.**

Please allow a warm-up time of at least 3 hours, preferably 6, for the instrument to settle any drift to within specification.



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

## 4.2 Connecting the QDAC-II to your experiment

It is advised not to connect your quantum devices to the instrument before power-up. You may even want to perform your own initialization after power up. 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  $\pm 10V$ ), but perhaps enough to cause damage to the connected devices.

## 4.3 Communicating with the instrument

**As mentioned in section 4.3, command strings should be terminated by a *New Line / Line Feed character (x0A)*.** This does not apply to binary blocks (see section 9). A *Carriage Return (x0D)* before the New Line character will be ignored and therefore does no harm.

Likewise, all responses, except for binary blocks sent by the instrument are terminated by a New Line character, according to the IEEE 488.2 standard.

### 4.3.1 Ethernet communication

When connecting the instrument<sup>2</sup> to a local network it will try to retrieve an IP address via DHCP. The network administrator should preferably lock an IP address to the instrument in the router. Alternatively, a fixed IP address can be set using the IPADdress command (see 4.3.1 and 7.6.1). If it is not possible to discover the instrument's IP address on the local network, it will be necessary first to connect via the USB/serial interface.

When opening a connection to the instrument it is important to use port 5025, otherwise the instrument will not respond. The instrument can have up to 8 concurrent TCP/IP connections.

Please remember to close TCP/IP connections properly after user, or the instrument may run out of connections (connections are reset when power cycling)

There is a latency of up to 10 ms when sending messages to the instrument over the ethernet port. Consequently, messages may become delayed up to 10 ms (and bundled with subsequent messages) even when connecting a computer directly to the instrument's ethernet port using no router or switch.

### Ethernet via a terminal

The easiest way to test the connection to the instrument is to use a simple Telnet client.

---

<sup>2</sup> In the current firmware version (13-1.53), to make the instrument acquire an IP address via DHCP, the ethernet cable should be plugged in *before* powering up the unit, or it should be restarted after plugging in the cable (SYSTEM:REStart).

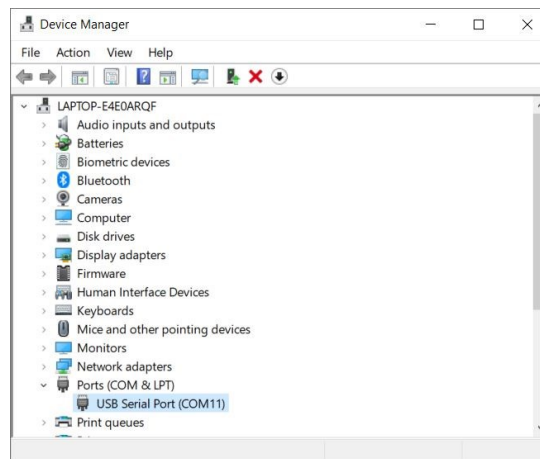
### 4.3.2 USB/serial communication

The instrument will appear as a serial device on the host computer. The communication settings are fixed and cannot be changed:

Serial port setting	Value
Communication rate	921600
Number of bits	8
Parity	None
Stop bits	1
Flow control	None

### 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 “COM11”.



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>.

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

### Control using a terminal program (Windows)

Several terminal programs are available for Windows for example CoolTerm (see *MAC OS and Unix/Linux* below). Another one is HTerm. The only requirement is that the program can send a NewLine character (also known as LF – linefeed) at the end of every message. An example of a program which will not do that is Putty; so it is recommended **not** trying to use that.



To install picocom please make sure that you have the Homebrew package manager installed (see <https://brew.sh>). Then install picocom using the command line

```
>brew -install
```

Starting picocom with the correct configuration. Please replace '/dev/cu.usbserial-1420' with the device address found above.

```
>picocom -b 921600 -c --omap crlf /dev/cu.usbserial-1440
```



```
ak — picocom -b 921600 -c --omap crlf /dev/cu.usbserial-1420 — 90x10
[> picocom -b 921600 -c --omap crlf /dev/cu.usbserial-1420
picocom v3.1

[*IDN?
QDevil, QDAC-II, A001234, 2-0.6.9
[SYST:COMM:LAN:IPAD?
"192.168.8.197"
```

CoolTerm (MAC OS, Linux, or Windows)

CoolTerm (GUI program) can be downloaded from [macupdate.com](http://macupdate.com) and is very easy to use.

Before connecting to the instrument, the Connection-Options have to be adjusted. Please update the port number with the previously found port ID:

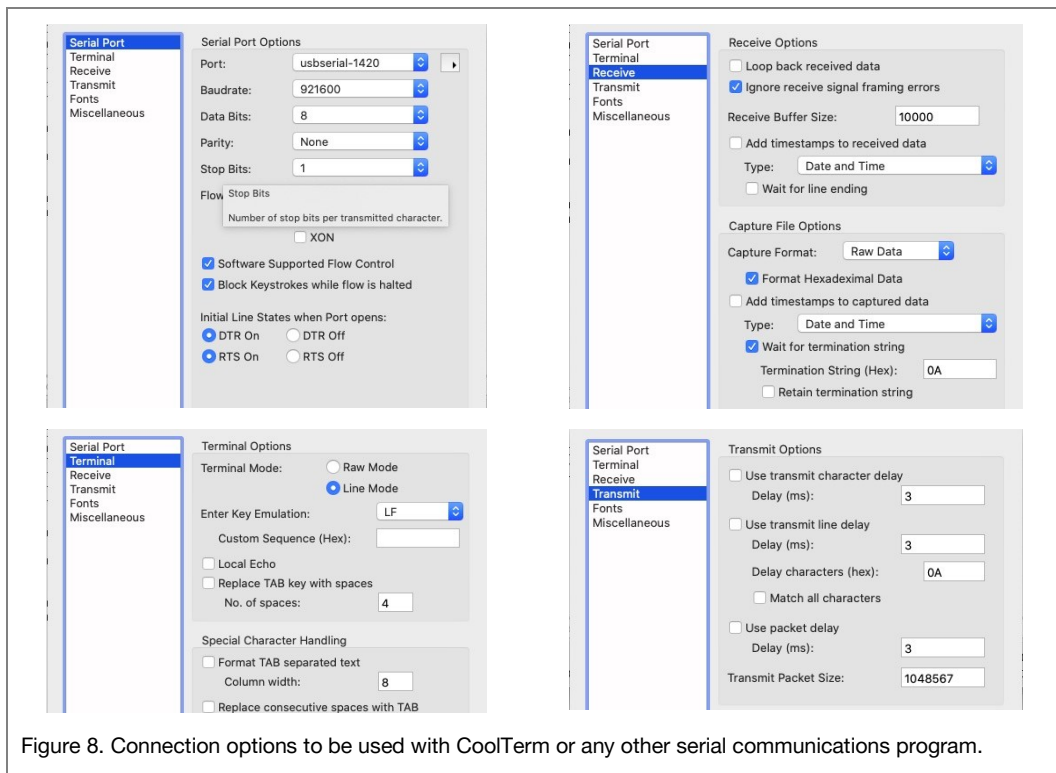


Figure 8. Connection options to be used with CoolTerm or any other serial communications program.

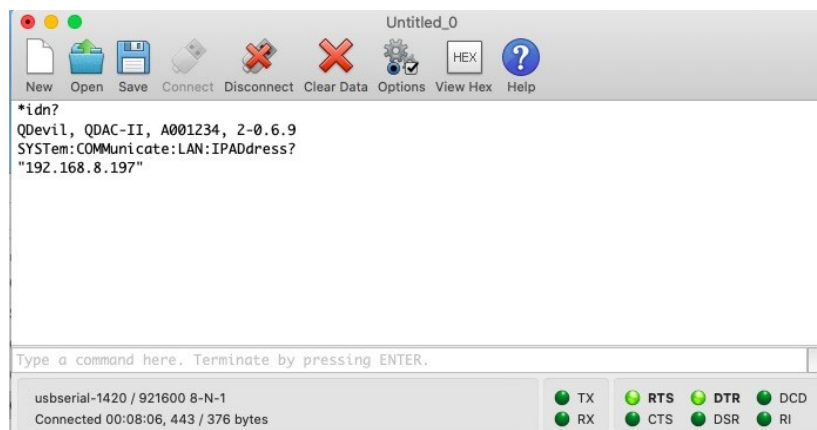


Figure 9. Example of sending a few commands via CoolTerm on a Mac.

## 4.4 Python

QDAC-II is intended to be controlled by a higher-level instrument control program. A driver is available for QCoDeS, which is Python based. Controlling the instrument directly from Python is also straight forward. One can connect to the USB/serial port using the *pyserial* package or to the TCP/IP ethernet connection using the *socket* package. However, often it is easier just to use the VISA library. To do so, the *pyvisa* package needs to be installed and imported. Further, either the ivi back-end (part of NI-VISA from National Instruments) or the *pyvisa.py* backend from the *pyvisa* consortium needs to be installed.

Especially when using the USB port, QDevil recommends using the IVI (NI-VISA) backend, as data may get garbled when big chunks are read back from the QDAC-II via USB with the *pyvisa.py* backend, see warning in section 5.2.3.

Once the above is all set a simple Python program, here using a Jupyter Notebook, may look like this:

```
import pyvisa as visa
class QDAC_II():
    def __init__(self, visa_addr="ASRL15:INSTR", lib=''):
        rm = visa.ResourceManager(lib) # To use pyvisa-py backend, use argument lib='@py'
        self._visa = rm.open_resource(visa_addr)
        self._visa.write_termination = '\n'
        self._visa.read_termination = '\n'
        # Set baudrate and stuff for serial communication only
        if (visa_addr.find("ASRL") != -1):
            self._visa.baud_rate = 921600
            self._visa.send_end = False
    def query(self, cmd):
        return self._visa.query(cmd)
    def write(self, cmd):
        self._visa.write(cmd)
    def write_binary_values(self, cmd, values):
        self._visa.write_binary_values(cmd, values)
    def __exit__(self):
        self.close()
```

```
rm = visa.ResourceManager('@py') # To use pyvisa-py backend, use argument '@py'
# List connected instruments. Instruments on LAN are often not shown.
rm.list_resources()
```

```
('ASRL15::INSTR',)
```

```
q = QDAC_II(visa_addr = "ASRL15::INSTR", lib = '')
# To use the ethernet port please replace visaAddress by the appropriate
# address in the form of: visaAddress = "TCPIP::192.168.8.197::5025::SOCKET")
```

```
print(q.query('*IDN?'))
print(q.query("syst:err:all?"))
```

```
QDevil, QDAC-II, 48762, 2-0.8.6
0, "No error"
```

```
# Start a 20kHz sine with 1V pp on ch1
q.write("sour1:sine:freq 20000")
q.write("sour1:sine:span 1")
q.write("sour1:sine:count inf")
q.write("sour1:sine:trig:sour IMM")
q.write("sour1:sine:init")
```

```
# Stop the sine generator
q.write("sour1:sine:abort")
# Set a DC voltage of 0.2 V on ch2 - ch5
q.write("sour:volt 0.2,@2:5")
```



#### 4.4.1 QCoDeS (Python)

QDevil has developed a QDAC-II (“QDAC2”) [driver](#) for QCoDeS, which is included in the [Qcodes\\_contrib\\_drivers](#) package in the official QCoDeS repository. Example Jupyter notebooks can be in the [docs/examples sub folder](#). Documentation can be found here: [https://qcodes.github.io/Qcodes\\_contrib\\_drivers/examples/QDevil/index.html](https://qcodes.github.io/Qcodes_contrib_drivers/examples/QDevil/index.html)

#### 4.4.2 Labber (Python)

At the date of publishing this manual a minimalistic Labber driver, is provided. The driver supports only DC voltages, including hardware ramping, and current measurement. Improvements are planned. Please find a link on the QDevil download page (<https://qm.quantum-machines.co/87kjeif6>).

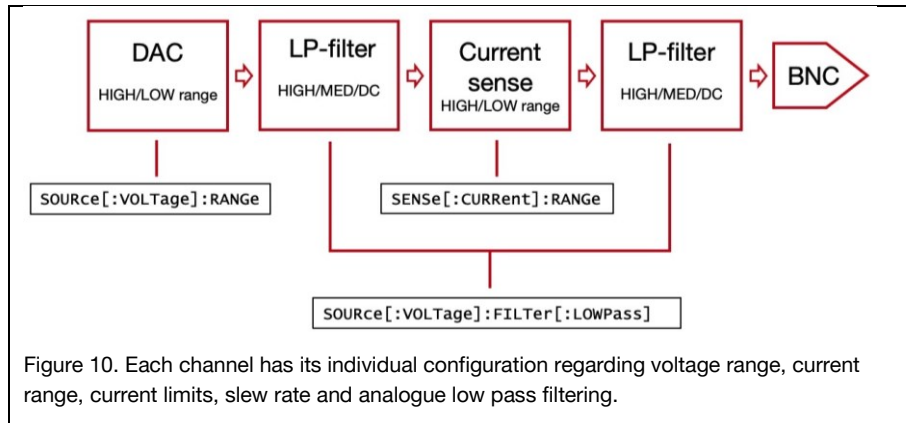
This driver consists of a driver definition file, [QDevil QDAC-II.ini](#), and a Python file with just a few chunks of code: [QDevil QDAC-II.py](#).

These two files should be placed in a subfolder called for example “QDevil\_QDAC-II” 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.

## 5 Features and functionality

### 5.1 Channel configuration

Each channel has global settings controlling its overall properties: Voltage range, current range and bandwidth:



Except for the current sensor, each channel is addressed by the keyword “SOURce” with the channel number appended. Addressing channel 14 this becomes “SOURce14”. For addressing multiple channels the “channel” SCPI Syntax can be used. Here we address channels 14 to 16: “SOURce<:other keywords> (@14:16)”. For more about SCPI syntax, see (see section 6.1). In the following we omit the channel(s) in the command descriptions. The current sensor is addressed using the “SENSe” keyword.

#### 5.1.1 Voltage range selection

The voltage range can be changed separately on each channel on the QDAC-II. On the standard product the ranges are  $\pm 10V$  (HIGH) and  $\pm 2V$  (LOW). The voltage range is set or queried by the RANGE command. Additional commands for querying the minimum and maximum values are also provided, primarily for use in drivers.

The advantage of using the LOW voltage range, if possible, is that it reduces the noise floor of the output, as the LOW range is generated as a voltage division after the digital to analogue conversion. In addition, the voltage resolution becomes 5 times higher, from nominally  $19.1 \mu V$  to  $3.8 \mu V$  at 20 bits DAC resolution.

Command	Description
<b>SOURce[:VOLTage]</b>	Node
:RANGe[?] {LOW HIGH}	Sets the output voltage range to HIGH ( $\pm 10V$ ) or LOW ( $\pm 2V$ ), or queries the currently selected range.
:RANGE:LOW:MINimum?	Queries the minimum achievable voltage in the LOW range.
:RANGE:HIGH:MINimum?	Queries the minimum achievable voltage in the HIGH range.
:RANGE:LOW:MAXimum?	Queries the maximum achievable voltage in the LOW range.
:RANGE:HIGH:MAXimum?	Queries the maximum achievable voltage in the HIGH range.

## 5.1.2 Low pass filters, output impedance

Each channel has low-pass filters with three choices of cut-off frequencies: DC, MEDium, HIGH.

Filter Setting	Nominal cut-off frequency	Resolution (bits)	Output capacitance
DC	10 Hz	25	1.0 $\mu$ F
MEDium	10 kHz	21	0.22 $\mu$ F
HIGH	230 kHz	20	10 nF

Table 1. Nominal RC low pass filter cut-off frequencies for the three selectable bandwidths. In addition, the effective bit resolutions and the RC filter output capacitances are listed. The series resistance in the RC filters is 50  $\Omega$ .

Note that there are two stages of low pass filtering on each channel (see Figure 4). Both stages are switched when changing the filter setting. The two filter stages have slightly different cut-offs and it is the lowest of the two which is given in Table 1. In addition to the application of the analogue low pass filtering, the filter setting also controls how much resolution enhancement is provided.

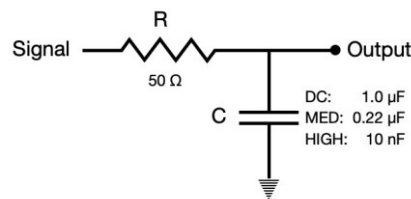


Figure 11. Output stage RC filter. The RC low pass to the BNC connector has  $R = 50 \Omega$ . The capacitance is determined by the which low pass filter is invoked. Note that the circuits connected to the BNC output may add capacitance thus lowering the effective low pass filter cut-off.

Command	Description
<b>SOURce[:VOLTage]</b>	Node
:FILTer[:LOWpass] {DC MEDium HIGH}	Sub-command for setting the lowpass filter cut-off for a channel

### Example

```
>SOUR:FILT MED, (@1:24) # Set the low-pass filter to MEDium for all channels.
```

### 5.1.3 Slew rate

In order to avoid sudden jumps in voltage, for example caused by mis-typing a DC value or a sweep span, the QDAC-II offers *slew-rate* limitation in hardware. That is a limit for how fast the voltage can change. So, the unit is Volts per second.

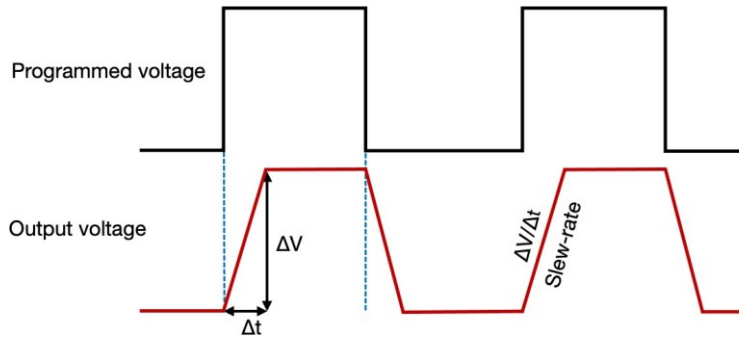


Figure 12. Slew-rate limitation sets an upper limit of how fast the voltage can change on the output of a channel.

Each channel has its own slew-rate settings. Further each voltage generator has its own individual setting. Often a low slew rate is requested on the DC generator, whereas a higher one must be used on the waveform generators, for example when overlaying a kilohertz sine. Note that the analogue low-pass filters will ultimately pose some limits for the highest obtainable slew-rate.

It must be noted that as generator signals are added the maximum slew rate will be the sum of slew-rates if high enough simultaneous voltages steps are applied on several generators.

Note that the slew-rate for waveform generators is the same for the AC part of the signal and for the DC offset.  
*A small transient (spike, negative or positive) must be expected to appear when changing the slew rate at non-zero voltage outputs.*

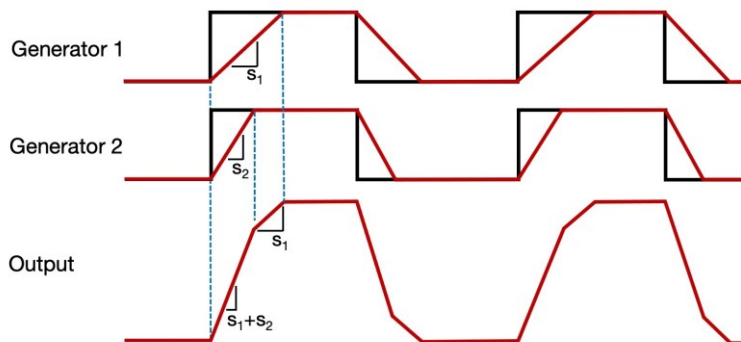


Figure 13. As the output of the five generators for each channel are added, the maximum possible slew-rate is a sum of the generator's individual slew-rate limits. This means that if fast voltage changes occur simultaneously on several generators the resulting voltage change rate may exceed the individually set limits.

Note that SLEW rate limitation should not be considered to be a substitute for using DC:SWEEp (fx in ANALog mode), as the SWEEp function has more options, in particular the ability of being stopped by the ABORt command. SLEWrate limitation works as if it was an integrating filter.

Command	Description
SOURce[:DC]:VOLTage:SLEW[?]	Sets/queries the slew-rate of the DC generator.
SOURce:[:VOLTage]:SLEW[?]	Sets/queries the slew-rate of waveform generators.

{ } = Sine, SQUare, TRIangle, or AWG. Note that setting the slew-rate for the DC generator implies that the is the same for all three modes (FIXed, SWEEp, LIST) of the DC generator.

### Examples

```
>SOUR2:VOLT:SLEW 115      # Sets the slew-rate of the DC generator on ch. 2 to 115 V/s
>SOUR24:SQU:SLEW 100     # Sets the slew-rate of the square wave generator on ch. 24 to 100 V/s
```

## Minimum and maximum limits for SLEWrate

The lowest slewrate possible is 0.01 V/s. The highest is  $2e7$  V/s (or *inf*), corresponding to a full range step of 20V in the time of one sample which is 1  $\mu$ s.

When FILTER = DC, and 25 bit resolution enhancement (RENHancement) is active, which it is by default, voltage generation is only possible using the DC generator which then has to be in FIXed mode. In this situation the lowest possible SLEWrate is 40 V/s. If the SLEWrate has been set lower than that, it will be clamped to 40 V/s until resolution enhancement is switched off, fx by exiting FILT = DC mode. A query will return the *set* value, not the *clamped* value.

### 5.1.4 Current range and current limitations

The QDAC-II has two current ranges. The measurement ranges are slightly lower than the actual sourcing limits. Nominally, the HIGH current range provides currents up to 10mA, and the LOW range up to 200nA. But, in a short circuit the instrument can actually deliver up to around +43 mA at +10 V and -51 mA at -10 V in the HIGH current range and around +4.3 mA at +10 V and -11 mA at -10 V in the LOW current range. *However, sourcing more than 10mA on just a few channels is not recommended*, as the internal power supply circuits are designed for all channels maximum sourcing 10 mA simultaneously. Further, to avoid stressing the current sensors one should preferentially avoid sourcing much more than  $\approx \pm 20$   $\mu$ A in the LOW current range.

The current sensing resolution is considerably higher in SENSE:RANGe:LOW range compared to the HIGH range. So LOW current mode is recommended for detecting leaks and measuring small current changes precisely. However, for fast signal generation the HIGH current range mode is recommended. The signal to noise level can be reduced when increasing the integration time (see section 5.3), though it should be noted that low frequency components (like 1/f) will remain.

Command	Description
SENSe[:CURRent]:RANGe	(Defines the current sensing range, HIGH (default) or LOW.

## 5.2 Voltage generation

As introduced in section 2, each channel is comprised of five voltage generators added together. The DC generator is considered the main generator, as the instrument is primarily a voltage source, where the four waveform generators are thought of as adding perturbations to the DC signal. So large sweeps are intended to be performed by the DC generator using either its SWEEP or LIST functionality. Except for setting an immediate DC voltage, the signal generators are all started using the trigger system, please see section 5.4 for details.

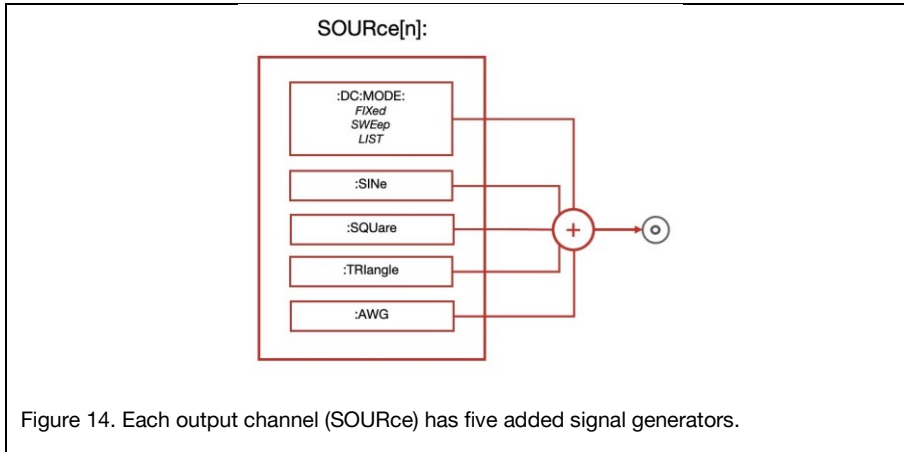


Figure 14. Each output channel (SOURCE) has five added signal generators.

Note that the final sum of generator outputs may differ from the actual output at the BNC connector as both the analogue lowpass filters and the load connected to the output will affect the practical slew-rate and rise time.

### 5.2.1 DC signal generation

The DC generator has three modes: FIXed, SWEEp, and LIST. In FIXed mode (default) the generator simply outputs a DC voltage. In SWEEp mode the DC generator produces staircase ramps where all steps have identical height and run (dwell). In LIST mode the DC generator outputs a user defined list of voltages either equally spaced in time or advanced one by one by trigger events. All three modes share the same slew-rate setting.

Note that FIXed mode can be thought of as a SWEEp or LIST with just a single point. Further, a SWEEp can be thought of as a special case of a LIST. Looking at it that way it is not so surprising that these modes share some commands, for example for reading the output of the DC generator.

Command	Description
SOURCE[:DC]	(top node)
[:VOLTage]:MODE	Sets or queries the functional mode of the DC generator (FIXed, SWEEp or LIST). Default is FIXed mode.
:VOLTage:SLEW	Sets or queries the maximum slew-rate for the DC generator.

:VOLTage[:LEVel[:IMMediate[AMPLitude]]?]	Queries the actual generated DC value at the time of the query. This may be slightly different from the set value due to the finite resolution of the D/A converters and if a finite slew-rate is set.
:VOLTage [:LEVel[:IMMediate[AMPLitude]]]:LAST?	Queries the last set DC value, see example under FIXed mode.

## FIXed mode

A DC output can be set immediately or at the next trigger event from the DC generator's trigger source. This way it is possible to simultaneously alter the DC output of multiple channels or synchronize the DC voltage change with external equipment such as for example RF generators, detectors, or sensors.

Command	Description
SOURce[:DC]:VOLTage	(top node)
[:LEVel[:IMMediate[AMPLitude]]]	Sets an IMMEDIATE DC voltage in FIXed mode. For query, see above.
[:LEVel]:TRIGger[:AMPLitude][?]	Sets or queries the DC value which will be set at the next trigger event in FIXed mode.
SOURce[:DC]:DAC[:LEVel[:IMMediate[AMPLitude]]][?]	Rarely used command for setting the DC output to a specific DAC code in FIXed mode. This command is primarily used in the calibration procedure. Likewise, the DAC code can be queried.

### Example – setting a voltage immediately or triggered

```
>SOUR:VOLT 0, (@1:24)           # Sets the voltage of all 24 channels to zero.
>SOUR:VOLT:TRIG 1, (@1:8)       # Prepares ch. 1-8 for stepping to 1V output at next trigger.
>SOUR:DC:INIT (@1:8)           # Triggers ch. 1-8 to change their output voltage to the triggered value.
```

### Example – difference between :VOLT?, :LAST?, and :TRIG?

```
>SOUR1:VOLT:SLEW 20             # Set the DC slew-rate of channel 1 to 20 V/s.
>SOUR1:VOLT 5;VOLT:TRIG 10      # Set an immediate output of 5V and the next triggered voltage to 10V.
wait 0.1 seconds                # Wait 0.1 second.
>SOUR1:VOLT?;VOLT:LAST?;VOLT:TRIG? # Simultaneously query the actual output, the programmed output and the next triggered output .
2;5;10                          # After 2 seconds have elapsed the output is 2 V as the slew-rate is 20V/s. The LAST set output is 5V and the next triggered output remains 10V. Note that due to a bug in current firmware version (13-1.57) LAST? Will in this case return 10 instead of 5...
```

## SWEep mode

In SWEep mode the instrument generates staircase sweeps between two voltages defined by START and a STOP. Besides the start and end voltages the staircase is defined by the time dwelled at each step, and the number of steps. The number of steps, equal to the number of voltage levels is given by POINTs.

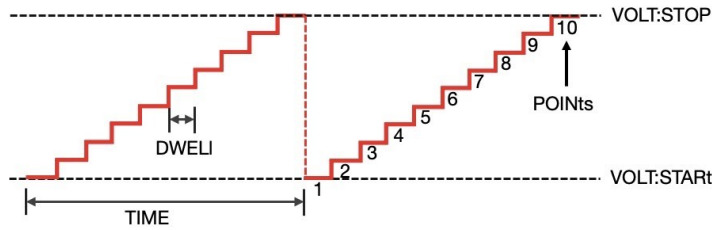


Figure 15. Main parameters for a stepped (staircase) DC generator sweep.

The time it takes for a single stepped sweep to complete is  $POINTs \times DWELI$ . The height of each step is given by  $(STOP - START) / (POINTs - 1)$ .

The sweep is repeated COUNT times. During the sweeping it is possible query the number of remaining sweeps using the NCLeft (number of counts left) command.

In addition to generating a staircase, the SWEep option can also generate a continuous ramp, when GENERation is set to ANALog. Due to the finite sample speed and the finite bit resolution of the DA converters even an ANALog sweep will have finite, however small, steps. The duration (TIME) for an ANALog SWEep is, as for STEPped SWEeps, given by  $DWELI \times POINTs$ , even though these two quantities separately are ambiguous for ANALog SWEep. The actual dwell (step length) is always equal to the instrument's sample update time of  $1 \mu s$ .

Be sure to set the DC slew-rate high enough for a stable level to be reached at every step, and in case of repeated sweeps for the STARrt level to be reached comfortably when returning from STOP and starting the next sweep ( $> |STOP-START| / DWELI$ ). Also set the DWELI time several times higher than the time constant of the used low-pass filter.

Command	Description
SOURce[:DC]:SWEep	(top node)
:TIME?	Read the duration of a single sweep.
:DWELI	The dwell time at each voltage level in the sweep.
:DWELI:AUTO	Enables auto calculation of the dwell time.
:POINTs	Number of voltage levels in the sweep, including the first and the last.
:COUNT	Number of repeats of the sweep.



:NCLeft	Number of repetitions left including the ongoing sweep.
[:VOLTage]:START	The voltage level of the first step.
[:VOLTage]:STOP	The voltage level of the last step.
:GENeration	As default SWEeps are staircases with a STEPped structure. However, they can also be continuous when GENeration is set to ANALog.

### Example – setting up and starting a stepped sweep

```

>SOUR8:SWE:VOLT:STAR -0.1 # Sets the start voltage to -0.1 V for a sweep on channel 8.
>SOUR8:SWE:VOLT:STOP 0.2 # Sets the end voltage to 0.3 V.
>SOUR8:SWE:DWEL 0.001 # Sets the dwell time to 1ms.
>SOUR8:SWE:COUN 1 # Just make a single sweep.
>SOUR8:DC:SWE:GEN STEP # Set the sweep generator to stepped mode. Not really required as this is the default.
>SOUR8:MODE SWE # Set the DC generator to SWEep mode
>SOUR8:SWE:INIT # Start the sweep (assuming that trigger source is set to IMMEDIATE).

```

### LIST mode

In LIST mode a channel will, when triggered, output a user defined sequence of absolute voltages. The sequence can be repeated, however not indefinitely. SWEep is a special case of LIST, where all voltage steps (voltage difference between consecutive points) are identical. LIST is used when the autogenerated SWEep sequence is not sufficient for the application. One example could be when non-linear, e.g. logarithmic, sequences are required. Another case could be when it is necessary to overlay a slow varying background to series of staircases, e.g. when compensating for cross coupling in a 2D gate-electrode scan of a quantum dot device.

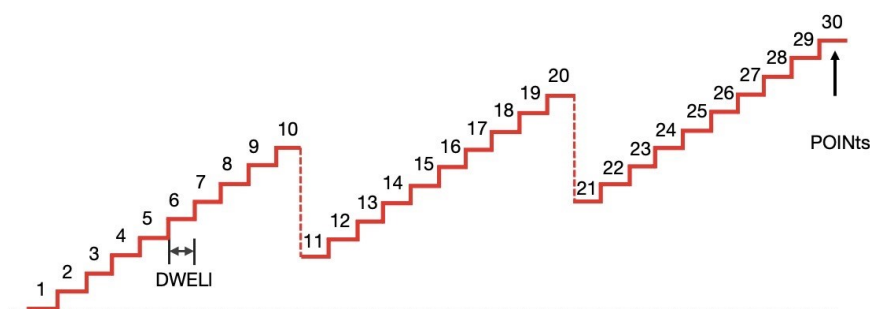


Figure 16. Example of 3 staircases in one LIST sequence each with an additional offset.

There also some similarities between LIST and the arbitrary waveform generator (AWG), but also some differences of which some are given in the table below. The AWG will typically be used for overlaying semi-fast curve shapes on to a DC signal, whereas LIST is used to produce a sequenced DC signal.

LIST	AWG
Variable playback speed	Fixed fast playback speed
One LIST per channel	Waveforms can be shared between channels
Absolute voltages	Scalable/offset-able numbers
Markers at each point, repetition, and start/end	Markers at start/end

As LIST sequences can be long, it is possible to choose between two formats when transferring the sequence of voltages using the SOURce:DC:LIST:VOLTage and :VOLTage:APPend commands: Ascii or binary block. The binary block format is described in section 9, and examples are provided in the command descriptions in section 7.3.2

Command	Description
SOURce[:DC]:LIST	(top node)
:COUNT	Number of iterations of the LIST, primarily relevant when TMODE = AUTO.
:DIRection	Determines the direction of traversing through the LIST sequence.
:DWELl	The dwell time at each voltage level in the LIST sequence (when TMODE = AUTO).
:NCLeft	Number of repetitions left including the ongoing iteration.
:VOLTage	Defines the voltages in the sequence (erases a previous sequence), in ascii mode only 1023 values are allowed. Use APPend for longer lists.
:VOLTage:APPend	Appends more points to the sequence (max. 1024 values at a time).
:VOLTage:POINts	Number of voltage levels in the LIST, including the first and the last.
:TMODE	Defines whether the complete LIST should run when triggered or if a trigger should just advance to the next point (first trigger "advances" to the first point).

### Example – setting up and starting a list

```
>SOUR8:LIST:VOLT 0,0.1,0.2,0.3,0.4,0.5,0.6 # Defines a sequence of voltages for the LIST on channel 8.
>SOUR8:LIST:VOLT:APP 0.7,0.8,0.9,1 # Adds additional points.
>SOUR8:LIST:DWEL 0.01 # Sets the dwell time to 10ms
>SOUR8:LIST:COUN 5 # Iterate the LIST sequence 5 times.
>SOUR8:LIST:TMOD AUTO # Sets the trigger mode such as to run the entire sequence on a single trigger.
>SOUR8:VOLT:MODE LIST # Sets the DC generator on channel 8 to LIST mode.
>SOUR8:DC:TRIG:SOUR IMM # Sets the trigger source to IMMEDIATE.
>SOUR8:DC:INIT # Initiates the DC generator and thereby starts it.
```

## Example – triggering the LIST point by point

```
>SOUR8:LIST:TMOD STEP                                     # Sets the trigger mode to STEPped so that each point in the sequence
                                                         # is advanced to by a separate trigger event.

For I = 1 to 5                                           # Pseudo code. Iterate through the 11 points in the LIST
  for j = 1 to 11                                         # sequence 5 times. Here INIT is used to make the IMM trigger
    >SOUR8:DC:INIT                                         # trig.
    # wait for settling
    # measure something
  end j
end i

>SOUR8:DC:TRIG:SOUR BUS                                  # The universal and preferred way is to use a "real" trigger event
>SOUR8:DC:INIT:CONT ON                                   # (which IMM is not) and fire that multiple times while INIT:CONT is
for i = 1 to 5                                           # ON. The global BUS trigger could be one example. But internal
  for j = 1 to 11                                         # triggers as well as external triggers are probably closer to
    >*trg                                                  # real life situations
    # wait for settling
    # measure something
  end j
end i
>SOUR8:DC:INIT:CONT OFF
```

## 5.2.2 Fixed waveform generators

Each channel has 3 fixed waveforms generators: SINE, SQUARE, and TRIANGLE.

The waveform generators have a lot of similar characteristics and commands which are described in the following paragraphs. In addition, all their commands are listed in separate paragraphs.

Note that waveform generators including AWG are only available in MEDIUM and HIGH filter modes, and *not* in DC filter mode.

### Common properties of waveform generators

For convenience it is possible to set a generator's frequency (FREQUENCY) as well as its period (PERIOD). The most recently set property will overwrite the other.

Waveform generators are by default applied symmetrically around the channel's DC value, as they are intended to serve as (small) perturbations to the DC generator signal. Their peak-to-peak amplitude is defined by the SPAN property. If a DC offset is required, this can be applied using the OFFSET property.

POLARITY specifies if the first part of a generator's period (or cycle) is positive (POS) or negative (NEG). If less than a 180 degrees phase shift is required this must be set using the DELAY property for the assigned trigger source.

The maximum slew rate (SLEW) can be set on an individual generator basis (default is INF). Note that it may be overruled by the channel's low-pass filter. The purpose of having finite slew rates for waveform generators is to restrict the edge-slopes for especially the square wave generator, but also to smooth the step applied when a generator starts or stops and its OFFSET is non zero.

By default a waveform generator continues indefinitely when started, until it is stopped by an ABORT command or when a property is changed. However, using the COUNT property it is also possible to set a specific number of cycles to run after starting a generator. Another property, NLEFT, (number of cycles left) can be queried to find out how many cycles are remaining once the generator has been started.

Generators are started using the trigger system, please see section 5.4.1. Interesting waveforms can be generated overlaying or coupling waveforms (and DC steps) using the MARKER and TRIGGER systems as the signals for all generators are added (see for example Figure 24). If the added signals extend the current voltage range, they will be clipped. No warning or error is produced when clipping occurs.

In order to offer as much flexibility as possible the allowable parameter ranges are not adjusted according to chosen slew-rates, low-pass filter, and the hardware imposed slew-rate limitation in for example the LOW current mode. A mismatch in parameters or exceeding the instrument capability may result in unexpected curve shapes.

**Warning: Asymmetric waveforms with frequencies above the filter cut-offs will impose a DC offset because the filter will average the AC signal! Note that square waves of only a few  $\mu$ s period will be highly asymmetric due to the asymmetric slew-rate of the output circuitry.**

## Waveform distortion and magic periods and duty-cycles

### *Discrete frequencies and periods*

Waveforms are generated by outputting a sample (a new voltage) every 1  $\mu$ s. This implies that only PERiods of an integer number of microseconds are possible. In other words, not all frequencies are allowed. The highest frequency which can be generated is 500 kHz (2 samples), then 333.33.. kHz (3 samples), then 250 kHz (4 samples) etc.

In the current version of the firmware (13-1.57) no error or warning is produced when trying to set the FREQuency or PERiod to an intermediate value. Instead, the period, or corresponding period when the frequency has been set, is rounded to the nearest integer number of microseconds. When queried, it will be the value set by the user which is returned and not the rounded value.

PERiod	FREQuency
2 $\mu$ s	500.00 kHz
3 $\mu$ s	333.33.. kHz
4 $\mu$ s	250.00 kHz
..	
33 $\mu$ s	30.30... kHz
..	
99 $\mu$ s	10.10.. kHz
100 $\mu$ s	10.00 kHz

Examples of supported periods and corresponding frequencies.  
Note: For TRIangle waveforms the minimum period is 4  $\mu$ s.

### *Magic periods and duty-cycles for accurate waveform generation*

The fact that a period must be an integer number of samples and due to the way that the fixed waveforms are implemented, a given waveform cannot be faithfully reproduced for all choices of periods and duty-cycles (square wave and triangle). Even for the sine waveform an integer number of samples are required per period, because otherwise it would not be possible to synchronize with the triggering system, which has the same 1  $\mu$ s update rate as the channel outputs.

The square wave can be defined by as little as two samples (each lasting 1  $\mu$ s). The output at that high frequency will look like a sinusoid with some DC offset, heavily damped by the low-pass filters. The next period giving a symmetric waveform is 4, and so on +2. For duty-cycles different from 50% it becomes a bit more subtle. A square wave with a period of 3  $\mu$ s will de facto have a duty cycle of 66.66.. % even if DCYCLE is set to 50 % (and hence is reported as 50%). For long periods the deviation will be small.

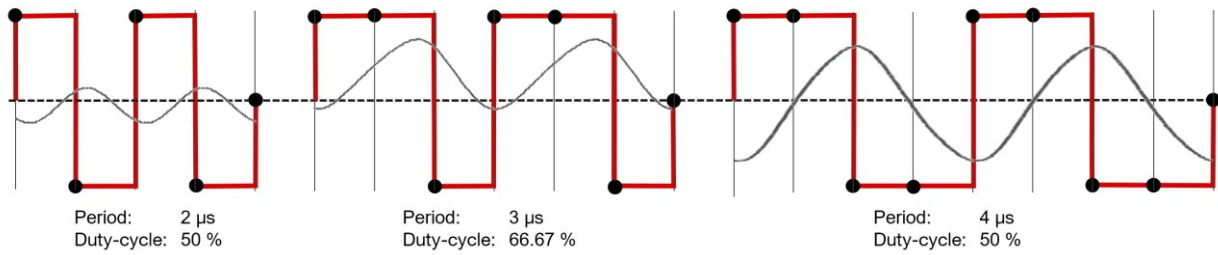


Figure 17. The three minimum periods for the SQUARE wave generator. Note that for the 3  $\mu\text{s}$  period, the duty-cycle will be 66.67 % even if it is set to 50%! The red curves are theoretical, the black dots representing every time a sample is output (every 1  $\mu\text{s}$ ). The actual output curves (HIGH filter mode) will be damped, and phase shifted by the low-pass filters, capacitive loads, and distorted/offset by the asymmetric finite driving force of the output amplifiers. Actual recorded curves are shown in grey color. Here it is clearly seen that asymmetric curves (e.g. 3  $\mu\text{s}$  period) will result in a net DC offset when the frequency is below or around the cut-off frequency of the low-pass filter.

The triangle waveform requires at least 4 samples (PERiod = 4  $\mu\text{s}$ ) for defining the two extrema and two zero points. The zero points (having start and ending points halfway between the extrema) are required for implementation purposes (and triggering). The next magic number of points is 8. So “perfect” symmetric triangles with 50 % duty-cycle are only available with periods which are multiples of 4  $\mu\text{s}$ .

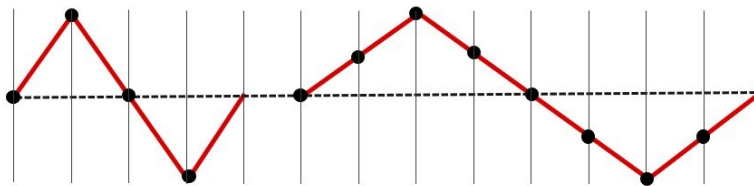


Figure 18. Number of points for generating the two shortest but faithfully reproduced triangle waveforms: Left: 4 points. Right: 4+4 points.

If the period is not a multiple of 4  $\mu\text{s}$ , the triangle will be approximately symmetric, but will end prematurely, meaning that there will be a small kink between repetition of periods, see Figure 19. For duty-cycles different from 50% it becomes a bit more subtle – each of the two waveform parts (up, down) should in addition be an integer number of microseconds for perfect waveform output.

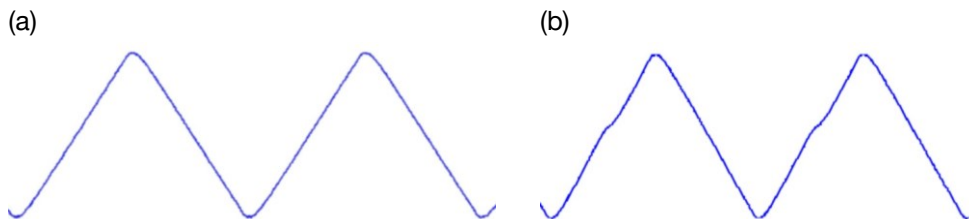


Figure 19. (a) Example of a “perfect” triangle waveform with a multiple of 4 points (28), and (b) one with a non “magic” number of points (26). A small kink is observed between repetitions of periods.

Note that regardless of these effect, **fast varying signals will be distorted due to the low-pass filter stages** and the finite (asymmetric) driving force of the output amplifiers, see section 10.2. Therefore, it may be difficult to observe the effect of magic/non-magic number of points for high frequencies / short periods. In the opposite regime, for very long periods (low frequencies), the relative distortion may be so small that it can hardly be observed.

## SINe generator

The sine wave generator is controlled in full by the common waveform generator commands.

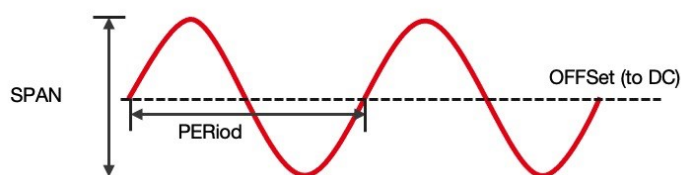


Figure 20. Basic properties of sine waves.

Command	Description
SOURce:SINE	(top node)
:PERiod	Sets or gets the waveform period in units of seconds (overwrites FREQUENCY)
:FREQUENCY	Sets or gets the frequency (unit of Hz) of the waveform (overwrites PERiod)
:COUNT	Determines the number waveform periods to be produced. Can be INFinite.
:NCLeft?	Number of repetitions left including the ongoing iteration.
:POLarity	Determines if the first half period is positive (NORMal) (default) or negative (INVerted).
[:VOLTage]:SPAN	Sets or gets the peak-to-peak voltage of the waveform.
[:VOLTage]:OFFSet	A (small) offset relative to DC can be applied.
[:VOLTage]:SLEW	Sets or queries the maximum slew-rate for this waveform generator.

### Sine wave generator limitations.

Each period of the sine is calculated with up to 65536 points and hence a very faithful reproduction of a sine is achieved, except at very long periods: As samples are output every 1 $\mu$ s this means that sines with periods longer than  $\approx$  65.5 ms, corresponding to frequencies smaller than about 15.38 Hz, will be interpolated. To produce a well-shaped sine waveform a certain minimum number of points (each 1  $\mu$ s) are needed. Together with the low pass filter this sets a limit for the upper usable frequency, e.g.  $\sim$  30.3 kHz. Note that a PERiod of 2  $\mu$ s will result in a flat curve, as the two calculated amplitudes will be the sine's two zero-crossing points. Also note that "magic periods" apply, see above, in order to have extrema and zero crossings represented. However, distortions are less evident than on the triangle.

### Example – program and starting a sine generator immediately

```
>sour8:filt high           # First set the filter, voltage range, and current sensor range for ch8 to HIGH
>sour8:rang high          in order to get the highest bandwidth. Note that, this is not necessary after
>sens8:rang high          power up as these are default settings.

>sour8:sine:freq 5000     # Then set the frequency to 5kHz, peak to peak amplitude to 4 volt.
>sour8:sine:span 4        # Set number of repetitions to infinite (default).
>sour8:sine:count inf

>sour8:sine:trig:sour imm # Set the trigger source to IMMEDIATE (default).
>sour8:sine:init          # Start the sine generator on ch8.
```

## SQUare wave generator

Both the square wave generator and the triangle wave generator share the concept of *duty-cycle*. For the square wave, duty cycle describes the percentage of each period taken up by the positive (first part) of the waveform, see Figure 21. In case POL = INVERTed, then it is the percentage of the negative part. With a 1  $\mu$ s sample output rate it is not possible to generate all combinations of periods and duty cycles. Please see the block about *Waveform distortion and magic periods and duty-cycles* above.

To make sure that the resulting curve looks as expected, always set the PERiod (or FREQuency) is set so that each part (low and high) contains an integer number of samples (of 1  $\mu$ s).

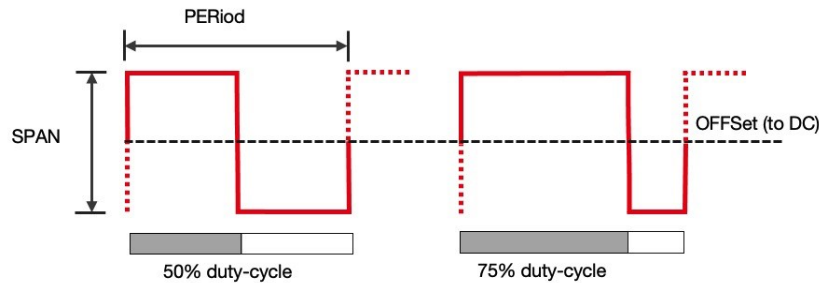


Figure 21. Basic properties of the square waveform. Note that the waveform always starts and ends with a voltage equal to OFFSet, defining a period. If the SLEWrate is low or MEDIUM filter is engaged, the flanks will be somewhat skewed.

Command	Description
SOURce:SQUare	(top node)
:PERiod	Sets or gets the waveform period in units of seconds (overwrites FREQuency)
:FREQuency	Sets or gets the frequency (unit of Hz) of the waveform (overwrites PERiod)
:DCYCLE	Duty cycle, the duration of the first half of the period divided by the entire period (default 50%).
:COUNT	Determines the number waveform periods to be produced (repetitions). Can be INFinite.
:NCLeft?	Number of repetitions left including the ongoing iteration.
:POLarity	Determines if the first half period is positive (NORMal) (default) or negative (INVERTed).
[:VOLTage]:SPAN	Sets or gets the peak-to-peak voltage of the waveform.
[:VOLTage]:OFFSet	A (small) offset relative to DC can be applied.
[:VOLTage]:SLEW	Sets or queries the maximum slew-rate for this waveform generator.



## TRiangle generator

For the triangle wave *duty cycle* describes the percentage of each period taken up by the positive going (first part) of the waveform. In case POL = INV, then it is the percentage of the negative going part. Please see the block about

*Waveform distortion and magic periods and duty-cycles* in the beginning of this section.

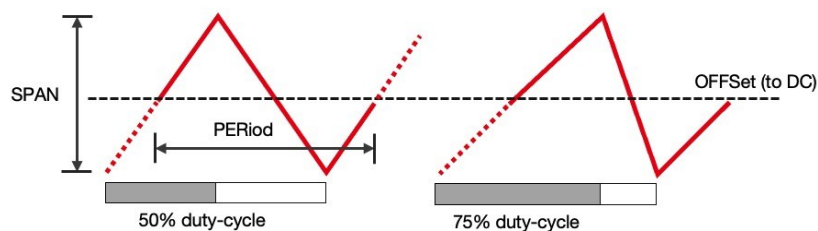


Figure 22. Basic properties of the triangle waveform. Note that the waveform always starts and ends with a voltage equal to OFFSet, defining a period. The duty cycle is best visualized by considering the period from one extremum to the next same type of extremum, e.g. from minimum to minimum.

Command	Description
SOURce:TRiangle	(top node)
:PERiod	Sets or gets the waveform period in units of seconds (overwrites FREQUENCY)
:FREQUENCY	Sets or gets the frequency (unit of Hz) of the waveform (overwrites PERiod)
:DCYCLE	Duty cycle, the duration of the first half of the period divided by the entire period (default 50%).
:COUNT	Determines the number waveform periods to be produced. Can be INFinite.
:NCLeft?	Number of repetitions left including the ongoing iteration.
:POLarity	Determines if the first half period is positive (NORMal) (default) or negative (INVerted).
[:VOLTage]:SPAN	Sets or gets the peak-to-peak voltage of the waveform.
[:VOLTage]:OFFSet	A (small) offset relative to DC can be applied.
[:VOLTage]:SLEW	Sets or queries the maximum slew-rate for this waveform generator.

### Example – setting up and starting a square wave with a triangle on top

```
>sour8:squ:freq 1000;span 2;trig:sour int1 # Defines square wave with e frequency of 1kHz and peak to peak
                                         # amplitude of 2V on channel 8. The trigger source is set to INTernal1.

>sour8:tri:freq 1e4;span0.2;trig:sour int1 # Defines triangle wave with e frequency of 10kHz and peak to peak
                                         # amplitude of 2V on channel 8. The trigger source is set to INTernal1.

>sour:squ:init                             # Both generators are initiated and started simultaneously by
>sour8:tri:init                             activating internal trigger 1.
>tint 1
```

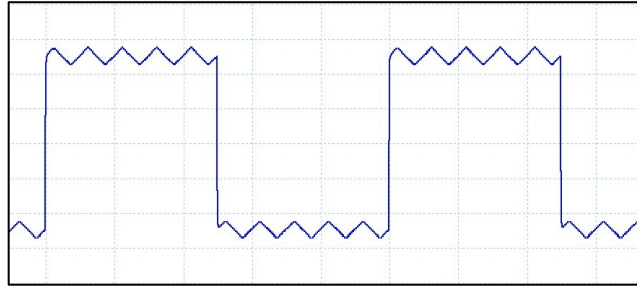


Figure 23. Scope trace of the added square and triangle wave signals. Note that the triangle starts and ends at its center value. If one wants to align the lowest point in the triangle with the positive edge of the square a `TRiangle:DElay` of a quarter of a triangle period can be specified.

### Example – Smearing the plateaus in a stepped sweep

```
>sour4:dc:volt:mode sweep                 # Define and start a stepped sweep (with just a
>sour4:swe:start -0.5;stop 0.5;dwe1 1e-3;poin 6;count inf # few steps and let it run indefinitely.
>sour4:dc:init

>sour4:dc:marker:ssstart:tnumber 7       # Make internal trigger no. 7 fire at the beginning
                                         # of every step.

>sour4:tri:per 50e-6;span 0.1;count 10   # Define a fast triangle with a just a few periods.

>sour4:tri:delay 0.2e-3;trig:sour int7   # Trigger the waveform when internal trigger 7 is
                                         # fired after a delay of 0.2 ms

>sour4:tri:init:cont on                   # Initialise the triangle generator in continuous
                                         # mode so that it can be retriggered indefinitely.
```

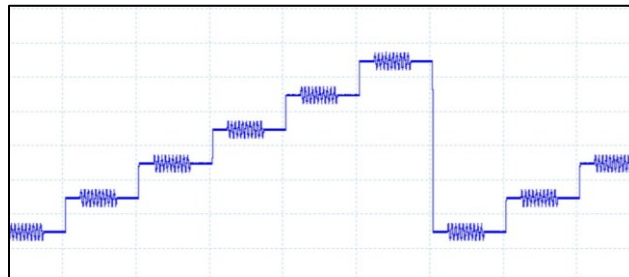


Figure 24. Scope trace of a stepped sweep with triangle waveform superimposed on every step. This is done in order to smear the applied voltage during the time of the step so that any measurements will be averaging over a small voltage range.

### 5.2.3 Arbitrary waveform generator (AWG)

The arbitrary waveform generator, of which there is one per channel, instead of generating waveforms on the fly, reads userdefined waveforms, TRACes, from QDAC-II volatile memory. The user uploads waveforms as TRACes, which can be “played” by the AWGs with individual scaling and DC offsets.

#### TRACes

Traces are blocks of normalized voltage data where each point corresponds to a sample of 1  $\mu$ s. Hence 1 million voltages correspond to a sequence of one second. The maximum length of a single trace is 6 million points (6,291,456). The length of traces should be an even number.

Note that with the current firmware version (13-1.57), no error is produced when uploading traces with uneven number of points. Instead, the last point is duplicated.

Trace values should be in the interval [-1;1]. Their scaling into actual volts is handled by the AWG generator of the relevant channels. The instrument has 1GB of memory for traces. In order to avoid timeouts, it is recommended to use TCP-IP communication (ethernet) when traces are more than around 40,000 points, especially when using the pyvisa framework.

A trace has three properties: a name, a length and its data points. The name can be a maximum of 15 characters long, length as mentioned maximum 6 mio. Datapoints are transferred as a binary block of 32 bit floating point values, see section 9. A maximum of 24 traces can be defined.

Command	Description
TRACe:	(top node)
:CATalog?	Returns a list of names of all traces.
:DEFine	Defines a trace by its name and number of data points.
:DATA	Transfers trace data to the instrument in binary format.
:REMOve:ALL	Clears the trace memory.

When traces become larger than a few tens of thousands points it is recommended to use TCP-IP communication rather than USB/serial. In both cases, when using frameworks with short default timeouts, it may be necessary to increase the timeout. For example in NI-VISA the timeout is by default just 2 seconds.

When communicating using VISA in Python, QDevil has good experience using both the pyvisa.py backend and the ivi backend included in NI-VISA.

However, when reading big amounts of data from the QDAC-II via USB, the data will often get garbled when using the pyvisa.py backend! So for USB the ivi backend is recommended.

On the other hand, when it comes to transferring traces the ivi backend (NI-VISA) has in rare cases been observed to time out unexpectedly.

### Example – Defining traces and viewing the trace catalog

```
>trace:remove:all # Deletes all traces in trace memory

>trace:define "PulsRCos20us",40 # Defines a trace of 40 points (40 μs)
>trace:define "Ring1ms",1000 # Defines a trace of 40 points (1 ms)
>trace:define "Ramp_nonlin_1s",1e6 # Defines a trace of 40 points (1 s)

>trac:cat? # Returns a catalogue (list) of all trace names.
"PulsRCos20us","Ring1ms","Ramp_nonlin_1s"
```

### Example – Uploading trace data using pyvisa (Python)

```
tracLen= 1000 # Generating a simple not very long trace as a simple Python list
p = 100/(2*math.pi)
values = []
for i in range(tracLen):
    values.append(math.sin(i/p)*(1-i/tracLen))

qdac2.write_binary_values \ # Transferring the data as a binary block using the pyvisa
("trace:data \"Ring1ms\"","values) "write_binary_values()" convenience function.
```

### The arbitrary waveform generator (AWG)

To “play” a trace on a channel, its arbitrary waveform should first be linked to the trace and, and the scaling and offset of the trace should be set (default 1 V and 0 V, respectively). In addition, the number of repetitions can be set as for the fixed waveform generators. Thereafter the AWG is started like any other generator using the trigger system. The same trace can be used on multiple channels simultaneously each with its own scaling and dc offset and trigger delay (start delay after triggering).

### Example – Assigning a trace to a channel’s AWG

```
>sour3:awg:define "Ring1ms " # Setup the AWG on channel 3 to play the “Ring1ms” trace with a voltage
>sour3:awg:scale 1;offset 0 scale of 1V and a zero dc offset, without repeating.
>sour3:awg:count 1

>sour3:awg:trig:sour imm # Set the trigger source to IMMEDIATE and make an IMMEDIATE INITIALIZATION
>sour3:awg:init in order to start the AWG on ch. 3 immediately.
```

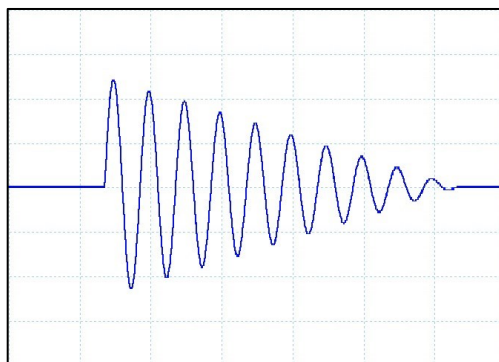


Figure 25. Scope trace of the “Ring 1ms” trace.

## 5.3 Current sensing

Each channel has an individually operating current sensor, each with two possible measurement ranges. See section 5.1.4 for commands.

### 5.3.1 Current measurement

#### Update rate and integration time

The current sourced out of a channel is probed by an IV converter. The voltage is converted into units of current at sampling intervals of 0.33.. milliseconds. However, the A/D converter applies a lowpass filter with a cut-off around 600 Hz, so the actual measurement speed is closer to 1-2 ms than to 0.33 ms.

To suppress noise, the sampled values are continuously integrated, and a running average calculated. The integration time is given by APERture. The most significant noise component is almost always induced 50 or 60 Hz mains power cycle noise. Therefore, it is wise to integrate over an integer number of power line cycles (PLC). APERture can be set indirectly by instead setting the number of PLCs to integrate over, using the NPLCycles command. The default (\*RST) number of PLCs is one, meaning that when the LFRrequency is 50 Hz, APERture will be 20 ms and for 60 Hz, 16.66.. ms. The maximum integration time is 2 seconds.

Whenever a measurement is triggered the currently integrated value is immediately transferred to the *measurement buffer*. This means that there is no wait time from a current measurement is requested until a value is delivered by the instrument. **It is up to the user to wait “long enough” between reads to accommodate changes in current caused by changes in the output voltages.**

The recommended minimum required waiting time is 3 ms for the shortest integration times (APERtures). For longer APERTures one should wait at least 3 ms + APERTure. This also means that when triggering current measurements (see further down) for example on a DC voltage change, one should use sufficiently long a trigger DELay.

Also, if the integration time (APERture or NPLC) is changed it is up to the user to wait sufficiently long time before reading out a new current measurement. **One needs to wait according to the difference in old and new integration time.**

If COUNT > 1 the instrument will space the values with APERTure seconds in time, but without waiting for the very first. Consequently, the reading will last (COUNT-1) x APERTure seconds.

#### Range switching time

When switching between the HIGH and LOW ranges, an electro-mechanical relay is toggling between two measurement resistors. This takes 30-40 milli-seconds. The SCPI command used to change the measurement range “SENSE:RANGe” will for that reason block the command execution for about 30 milli seconds. However, this does not prevent the user from queuing up more commands in the meantime. **If the current range is changed for N channels in a *compound command* [6.2], then there will be a delay of N x 30 ms before the next command is executed! This must be taken into account before doing the next current reading. In addition, one should wait for the duration of one APERTure after the range switching has completed.** A simple way of checking when the RANGe command is complete is to send a query to the instrument, for example a short one such as \*STB?.

## Example – changing ranges and apertures

```
>sens:rang low, (@1:5)           # Set the current range to low for channels 1 to 5.
>*stb?                          # Make a simple query to test when the range switching is complete.
0                                # Alternatively one can wait 5 x 30ms: sleep(0.15).
>sens:aper 0.2, (@1:5)         # Change the APERTure to 0.2 sec for channel 1-5.
sleep(0.2)                      # Pseudo code. Assuming that the APERTure was 20 ms before, sleep 0.2 s.
>read? (@1:5)                  # Readout all 5 channels
5.56377e-10,1.02779e-09,-5.01892e-11,
4.50601e-10,6.00684e-10
```

Note that, in addition to the above, it takes some time for any relay contact to stabilize completely when switched from *off* to *on*. Therefore, when switching from LOW to HIGH where there is a very small resistor value in series with the relay, it is recommended to wait a couple seconds (or more) before taking current readings in order to achieve stable readings.

Note that sending a command is not the same as it will be executed immediately. When using LAN communication there can be up to 10 ms delay from a command is present at the LAN connector until it is executed, which should be considered.

The safest way to ensure that the instrument has executed all sent commands is to perform a small query, for example `*STB?`, and then wait for the response.

## Current range: Implied voltage generation bandwidth

The in-line IV converter utilizes a sensing resistor of which there are two, one for the HIGH current mode and one for the LOW current mode. Due to the high value used in LOW current mode and due to the inherent non-linearities in the active circuits the effective bandwidth of the instrument in LOW current mode is not a simple RC filter behavior but is AC amplitude and DC voltage dependent. At DC voltages and AC peak-to-peak amplitudes smaller than about 3V, the behavior is linear, but at higher amplitudes or DC offsets the behavior is non-linear, limiting the available  $dV/dt$  (slew). The curves shown in section 10.2 in “10 Specifications” illustrate this non-linear behavior.

## Accuracy and precision

### Nonlinearity, and precision

Due to in particular the nonlinear output voltage dependence of the IV conversion (common mode error) the accuracy is limited by integral nonlinearity which is of the order of:

$$\text{Max int. nonlin. (LOW)} \approx 1 \text{ nA/V} \times (V_{\text{out}} - I_{\text{out}} \times 0.04 \text{ V/nA}) \pm 3.5 \text{ nA}$$

$$\text{Max int. nonlin. (HIGH)} \approx 0.1 \text{ mA/V} \times (V_{\text{out}} - I_{\text{out}} \times 0.001 \text{ V/mA}) \pm 0.35 \text{ mA}$$

Where  $V_{\text{out}}$  is the output voltage (as set) and  $I_{\text{out}}$  is the output current (measured). The first term can actually be used for compensating the recorded currents for the first order common mode effect.

This means that in LOW current mode, if no post correction is performed the maximum error is around  $1\text{nA/V} \times (10\text{V}-0\text{nA} \times 0.04\text{V/nA}) + 3.5\text{nA} \approx 14\text{nA}$ , assuming resistive loads. In HIGH current mode the corresponding number for is about 1.4mA. [Future firmware versions may incorporate this compensation.](#)

As described in section 3 (Current sensing resolution), the resolution is considerably better in LOW current mode than in HIGH current mode. Please see section 10.3 in *10 Specifications* for details and examples.

### Offset error

The current sensors have a non-negligible offset dependence on temperature, which is not compensated for in the current version of the firmware. Future versions may include some compensation. This means that if the temperature of the environment where the instrument is used is different from the temperature in the laboratory where the instrument was calibrated a non-zero value will be read out even when a channel is unloaded, and the voltage is zero. This error is typically around 1nA in LOW range and 0.1mA in HIGH current range but can be up to 5 times higher. It is therefore recommended to record the “zero current” before measurements, if the absolute value is of importance.

### Commands

The easiest way to measure current is to use the READ? query command, as it will initiate and trigger a single measurement (of COUNT values) and transfer the result to the output buffer. Note that if COUNT is large or APERTure is larger than the READ? query may time out, leaving values in the output buffer.

The measurement buffer can hold up to 65536 values. If more values are added, a memory overflow error will be produced.

To avoid timeouts or to synchronize current measurements with other events or equipment measurements can instead be triggered and held in the measurement buffer until they are queried. The trigger commands are listed below, but please see section 5.4 for details on the trigger system.

Commands for configuring, reading, and triggering current measurements.

Command	Description
SENSe[:CURRent]	(top node)
:ABORt	Stops on going measurements and trigger sequencies.
:APERture	Defines the averaging time in seconds (default one power line cycle).
:NPLCycles	Defines the averaging time in units of power line cycles (default 1).
:COUNT	Defines the number or measurements to perform on per trigger (default 1).
:NCLeft?	When performing more than one measurement (COUNT > 1) NCLeft will report how many measurements are left at a given time for the ongoing trigger cycle.
:DELay	Specifies the delay after a trigger signal is received until the measurement is performed
:INITiate[:IMMEDIATE]	Makes the sensor ready to receive a trigger and run a single trigger cycle, where after it will go back into idle mode. If the trigger source is IMMEDIATE, then a measurement is performed immediately.

:INITiate:CONTinuous	Switches continuous mode on or off. When set to ON the sensor will react to every trigger received until it is set to OFF.
:TRIGger:SOURce	Defines the trigger source (INTernal#, EXTernal#, BUS, IMMEDIATE). Default is IMMEDIATE
:RANGe	Defines the current sensing range, HIGH (default) or LOW.
:DATA:POINTs?	Queries the number of measurement points in the measurement buffer.
:DATA:REMOve?	Queries and deletes a specified number, or all measurement points in the measurement buffer.
:DATA:LAST?	Queries the most recent <i>triggered</i> measurement point. This can be done even if the measurement buffer has been emptied.
FETCH[:CURRent]?	Reads all data points in the measurement buffer, without clearing the buffer.
READ[:CURRent]?	Runs a single trigger cycle and returns the (COUNT) measurement datapoints. If the trigger sequence is initiated and waiting for a trigger, READ? will assume priority*.
READ:ADC?	Reads the raw code from the analog to digital converter used for current sensing. Mostly used during factory calibration. Reports a single value also if COUNT > 0.

\*)Note that the READ? Command will behind the scenes set the trigger source of the current sensor to IMMEDIATE, INIT:CONT to OFF and issue a single INIT:IMMEDIATE command. After the read command INIT:CONT will be off and TRIG:SOUR will be IMMEDIATE. So, any triggered operation will as a consequence be aborted.

### Example – read a single current value

```
>READ8?  
2.53123e-3
```

*# Captures the and returns the current value of the running average on channel 8 (2.53. mA).*

### Example – trigger and read 3 measurements on ch4

```
>sens4:coun 3  
>sens4:trig:sour imm  
>sens4:init  
>sens4:data:poin?  
3  
>fetch4?  
-0.0010562,-0.0010562,-0.0010563  
>sens4:data:rem?  
-0.0010562,-0.0010562,-0.0010563  
>sens4:data:poin?  
0
```

*# Set the number of readings to 3  
# Set the trigger source to immediate (default).  
# Start one trigger cycle.  
# Read the number of points in the measurement buffer.  
  
# Read the entire measurement buffer for the channel with our clearing.  
  
# Read and remove all datapoints in the measurement buffer.*



## 5.4 Trigger system

The QDAC-II has an advanced triggering system allowing synchronization of programmed changes in for example DC steps and waveforms.

Except for a few global commands there is trigger system with one sequence per channel functionality: Under each source:<source name> node there will be an independent trigger sequence. In addition, there is a trigger sequence for the current measurement on each channel; please see section 5.3.1 for details.

The trigger model is a very simplified version of the SCPI model, omitting the arm layer.

Starting and stopping generators is done by the generator's trigger system. All generators have an identical trigger system (see next section). and all generators can be addressed using the source name "ALL", e.g. "SOUR:ALL:INIT".

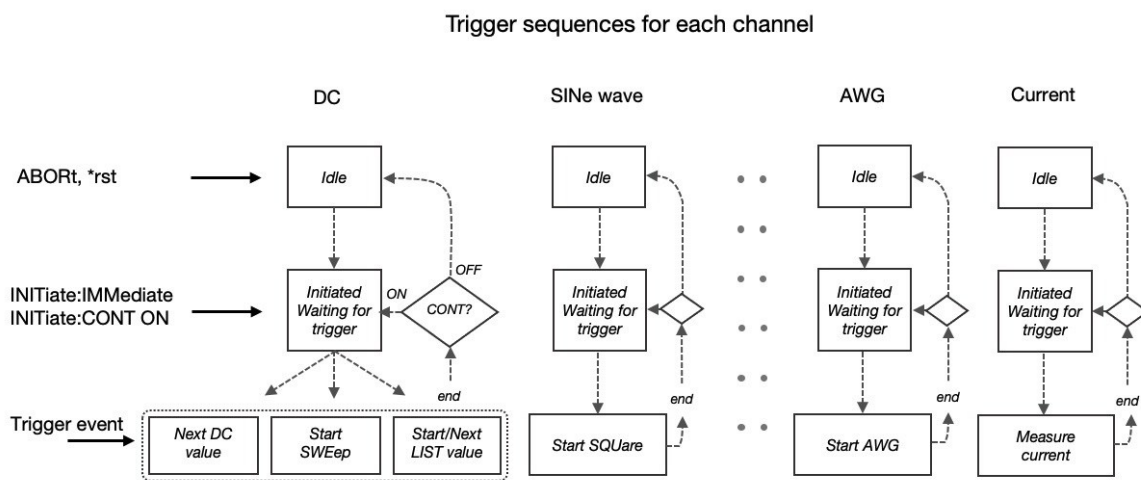


Figure 26. Trigger model. Every channel has a trigger sequence for each of its sources and for the current measurements.

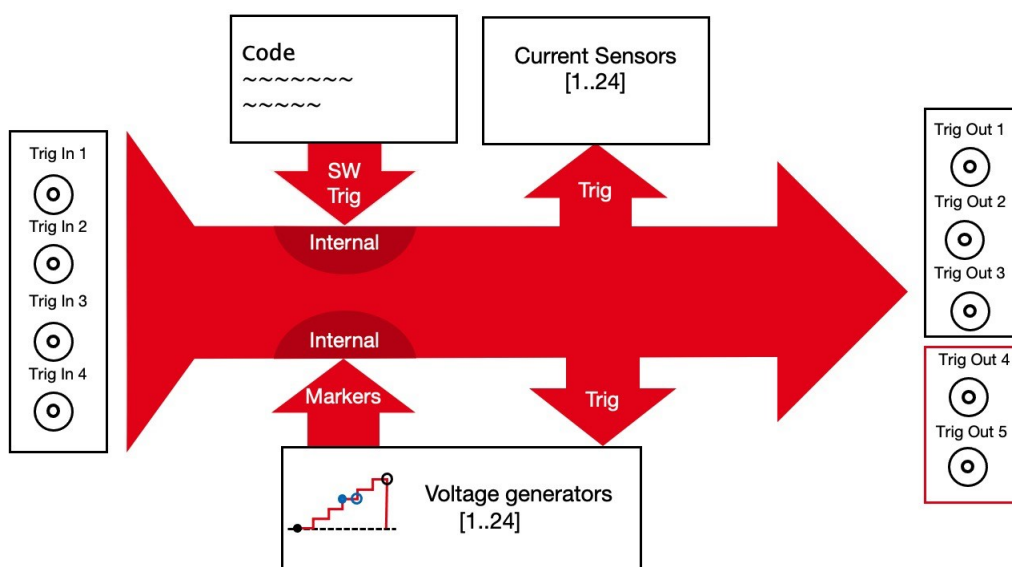


Figure 27. The QDAC-II trigger system allows synchronization between channels and waveforms and between multiple QDAC-II units. There are 14 internal triggers available.

### 5.4.1 Starting (triggering) and stopping voltage generators

Except for the simple case of just setting a DC voltage all generators need to be started by a trigger event. The reason is that each generator has several parameters which one would like to set correctly before run. Further triggering provides many ways of configuring coordinated waveform generation and measurements.

It is, however, very simple to trigger a generator as all have their default trigger source set to IMMEDIATE, meaning that a trigger signal is always assumed to be present; all what is needed is to INITiate the generator, and it will start. Trigger sources can also come from the external trigger inputs the internal BUS or the *internal triggers* which can be connected to signal generator *markers*-

For examples of starting/triggering generators, please see section 5.2.

Command	Description
SOURce:{DC SINe SQUare TRIangle AWG ALL}	(top node)
:ABORt	Stops the specified generator.
:DELay	Specifies a delay from the trigger signal until the generator actually starts.
:INITiate	Makes the generator wait for a trigger signal (if the trigger source is Immediate, which is the default, this command will start the generator).
:TRIGger:SOURce	Specifies the trigger source.

### 5.4.2 Triggering of current sensors

Current sensors are triggered in the same way as voltage generators with the commands described in section 5.3.1, where examples can also be found. The current sensors themselves cannot generate trigger signals. As their timing is known it is not necessary.

### 5.4.3 Trigger Outputs and Inputs

The QDAC-II has three galvanically isolated  $\approx 5$  V trigger outputs on the front panel and two *non-isolated* 3.3 V outputs on the rear panel. The output impedance of the outputs on the front panel is nominally 50  $\Omega$  ( $\pm 20$   $\Omega$  uncertainty). The output impedance of Trig Out 4 and 5 is around 30  $\Omega$ .

The five Trigger Outputs are either controlled by internal triggers, external trigger inputs or software triggers and commands.

Note that the trigger output pulses appear approximately 1  $\mu$ s *before* the marker position of the voltage generator output due to different delays (phase shifts) in the analogue circuits. To compensate, a suitable trigger DELay can be added (1  $\mu$ s increments).

The QDAC-II has 4 galvanically isolated trigger BNC connector trigger inputs on the rear panel. These are used for synchronizing the instrument to external devices or other QDAC-II units. The trigger pulses must be at least +2.2 V (preferably > +2.5 V) high. Negative voltages (below -0.5 V) should be avoided not to damage the input circuits. Further, they should not exceed +3.8 V. This is particularly important when connecting an external clock signal to Clock input (*Trigger In 4*) because the *on*-time will typically be 50%, and not just occasional pulses.

For the 3.3 V trigger outputs to drive these inputs it is recommended to use 75  $\Omega$  cabling for interconnection multiple QDAC-IIs. This will give a slightly higher signal compared to using 50  $\Omega$  cabling, as it is highly recommended to terminate the interconnecting cable according to the impedance.

There are two reasons for terminating the cables:

1. Standing waves will alter the signal quality seen by the respective instruments in the chain.
2. Standing waves may increase the signal height above the allowed 3.8V or make it lower than - 0.5 V.

Commands for configuring and connecting Trigger Outputs to internal or external triggers

Command	Description
OUTPut:TRIGger	Top node
:SOURce	Specifies which input or internal trigger should control the Trigger Out signal
:WIDTh	Specifies the width of the trigger output pulse, minimum 1 $\mu$ s.
:POLarity	Specifies if the pulse is positive or negative
:DELay	Specifies a delay from receipt of a signal from tegh SOURce until the trigger pulse is output
:SOURce	Specifies the source (INTernal1..14, EXTernal1..4, BUS, HOLD)

### Example – Output a 10 $\mu$ s pulse on Trig Out 1 when a sweep starts

```
>sour1:dc:swe:poin 4096                # Set up the sweep on ch1
>sour1:dc:swe:dwell 0.0001
>sour1:dc:swe:count 1
>sour1:dc:swe:star -0.2;stop 0.6
>sour1:dc:volt:mode sweep
>sour1:dc:trig:sour IMM
>sour1:dc:marker:start:tnum 1          # Connect the sweep period start marker to internal trigger 1
>outp:trig1:sour int1                 # Make Trig Out 1 listen to internal trigger 1
>outp:trig1:width 0.0001             # Set the pulse duration or Trig Out 1
>sour1:dc:init                        # Start the dc generator onch1 by initializing it (trigger source = immediate)
```

## 5.4.4 Internal triggers and Markers

### Internal triggers

*Internal triggers* can start a generator or trigger the next step in a sweep or list sequence or trigger a current measurement. Internal triggers can also trigger external devices when it is routed to one of the Trigger Out connectors. *Internal triggers* can be fired programmatically using a scpi command (`TINT`) for example for starting several generators simultaneously on the same or on multiple channels, or they can be fired by *marker* events, see below. There are 14 internal triggers available.

All generators have a configurable trigger *DELay* so that starting can be delayed until an arbitrary chosen time (in 1 $\mu$ s increments) after the trigger event.

The standard scpi global `*trg` command is an internal “master” trigger which, however, cannot be activated by markers.

## Markers

Markers are events on waveforms and DC sequences which can activate *internal triggers*.

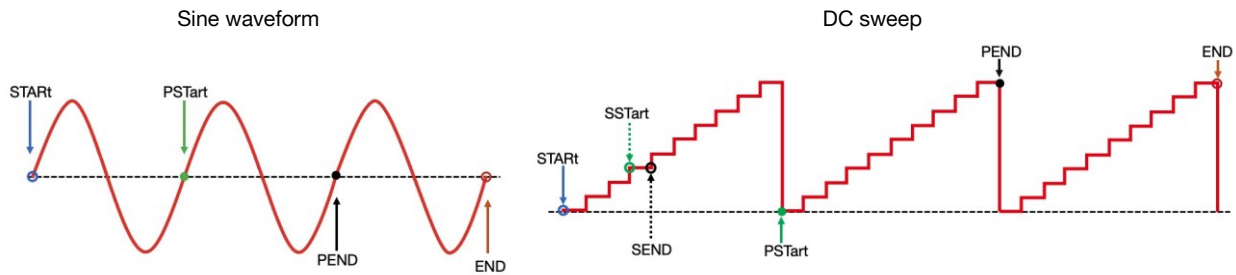


Figure 28. Waveforms and DC sequences have *markers* associated with them. One pair marking the start and end of the waveform or sequence, and one pair marking the start and end of every period. The PSTart and Pend markers shown above are actually called SStart and SEND for DC LIST and SWEEp and mark the beginning and the end of a voltage step.

There are four markers for each generator, six for the DC generator. For the sine, square, triangle and AWG there are markers at the start and at the end of the signal (START, END), and at the start and end of each period (PSTart, PEND). The end of period marker is in general only used in those rare cases where a delay of one period from the start is needed.

For the DC generator there are also step-start and step-end (SStart, SEND) markers for the LIST and SWEEp modes generating markers at the start and end of each step. For ANALog SWEEps these markers coincide with the PSTart/PEND markers.

Note that the START and PSTart markers will both be present at the beginning of the waveform. In the same way the END and PEND markers will both be present at the end of the waveform. For the DC generator the same counts for the SStart and SEND markers.

Commands connecting generator markers to internal triggers

Command	Description
SOURce:{ } :MARKer { } = DC: SINE TRIangle SQUare AWG ALL	Top node for markers
:STARt:TNUMBER	Specifies which internal trigger is linked to the STARt marker.
:END:TNUMBER	Specifies which internal trigger is linked to the END marker.
:PStart:TNUMBER	Specifies which internal trigger is linked to the <i>period start</i> (PStart) marker.
:PENd:TNUMBER	Specifies which internal trigger is linked to the <i>period end</i> (PENd) marker.
SOURce:DC:SStart:TNUMBER	Specifies which internal trigger is linked to the SStart (step start) marker.
SOURce:DC:SEND:TNUMBER	Specifies which internal trigger is linked to the SEND (step end) marker.
TINT[:SIGNal]	Immediately fires the specified internal trigger.

### Example – Start sine generator on ch11 when the sine on ch3 ends.

```
>sour:sine:freq 7500,(@3,11)      # Set the frequency of the sine generators on ch3 and ch11 to 7.5 kHz.
>sour:sine:span 0.2,(@3,11)      # Set peak-to-peak amplitude of the sine on ch3 and ch11 to 200 mV.
>sour:sine:coun 75,(@3,11)       # Set the number of periods per run for each sine to 75.
>sour3:sine:mark:END:TNUM 1       # Connect internal trigger 1 to the END marker for the sine wave on ch3.
>sour11:trig:sour INT1           # Set the trigger source for the sine on ch11 to internal trigger 1
>sour11:sine:init                # Initiate the sine generator on ch11 so it is ready to receive a trigger.
>sour3:trig:sour IMM             # Make sure that the trigger source for the sine generator on ch3 is
                                # immediate.
>sour3:sine:init                 # Start the sine generator on ch3
```

### Example – Start sweeps on ch1 and ch2 simultaneously using internal triggering

```
>sour:dc:swe:poin 128,(@1,2)     # Set number of ramp steps on ch1 and 2 to 128
>sour:dc:swe:dwell 0.05,(@1,2)  # Set duration of each level to 50 ms
>sour:dc:swe:count 1,(@1,2)     # Set number of ramps to one
>sour1:dc:swe:star -0.1          # Set start and stop voltages for the two sweeps
>sour1:dc:swe:stop 0.3
>sour2:dc:swe:star 0
>sour2:dc:swe:stop 1.2
>sour:dc:volt:mode sweep,(@1,2) # Set the dc generator on ch1 and ch2 to sweep mode
>sour:dc:trig:sour INT1,(@1,2)  # Set the trigger source on both channels to internal trigger 1
>sour:dc:init (@1,2)            # Initialise the dc generator on both channels
>tint 1                          # Activate internal trigger 1 to start the two seeps.
```

## 5.5 Synchronization of multiple QDAC-II units

Multiple QDAC-IIs can be inter-connected so that samples are output synchronously, meaning that ramps, waveforms and DC values can be triggered to appear simultaneously to within fractions of a microsecond. Note that all cables should be terminated according to their characteristic impedance, typically 50  $\Omega$  or 75  $\Omega$ .

It is possible to use an external 10 MHz clock for all units, but one QDAC-II unit can also act as *reference* and generate a shared 10 MHz clock signal. 10 MHz frequency is chosen for compatibility with other instruments.

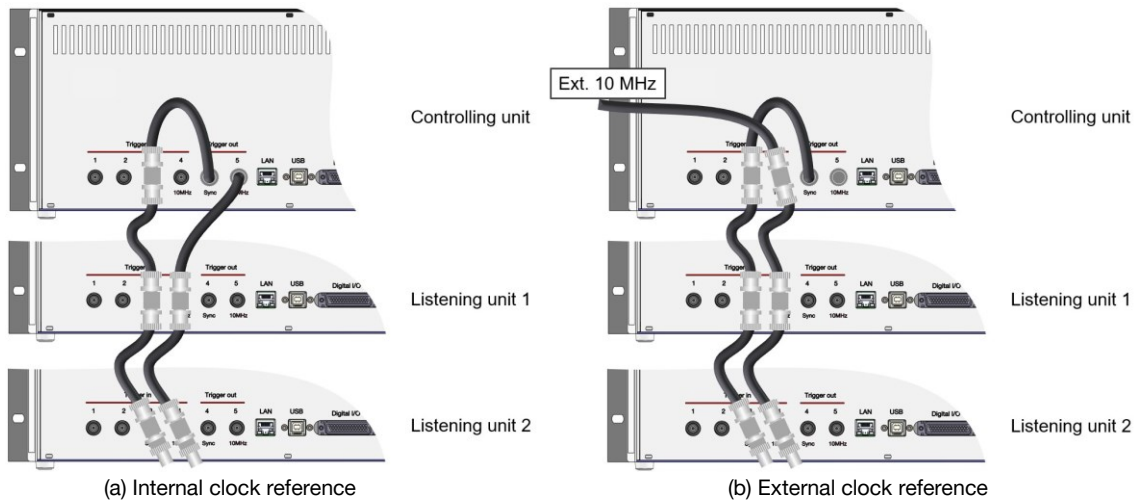


Figure 29. Wiring up three synchronized QDAC-IIs, with (a) use of 10 MHz clock generated by the master unit, and (b) external 10 MHz clock. Note that both cables need to be terminated according to their impedance.

Voltage samples are output at a rate of 1 MHz. The 1 MHz clock is generated from the 10 MHz internal or external clock. To make sure that the 1 MHz clocks on connected units sharing the same 10 MHz clock are in phase, they need to be aligned. Otherwise, they may be phase shifted by any multiple  $<9$  of 100 ns.

Alignment is carried out by a one-time *synchronization* of the units upon a positive edge at the *Sync In* port (*Trigger In #3*). One QDAC unit needs to be the controller and output a Sync pulse on its *Sync Out* (*Trigger Out #4*) port which should therefore be routed to the *Sync In* ports of all units including the controller itself. An external pulse can also be used (see section 3 “Rear panel” for appropriate voltage levels).

When waiting for the signal on *Sync In* the instruments will enter *Sync Wait* mode and will freeze their outputs for a short period of time. There is a time-out limit of 2 seconds in waiting for the *Sync In* signal.

Note that the use of terminated 75  $\Omega$  cabling is recommended in order to ensure that the *On* thresholds of the Trigger Inputs are exceeded. Also, note that BNC cables have a group delay of about 5 ns/m, which may influence the timing. Typical timings are better than 50 ns, often in the range of 20-30 ns using short impedance matched cables.

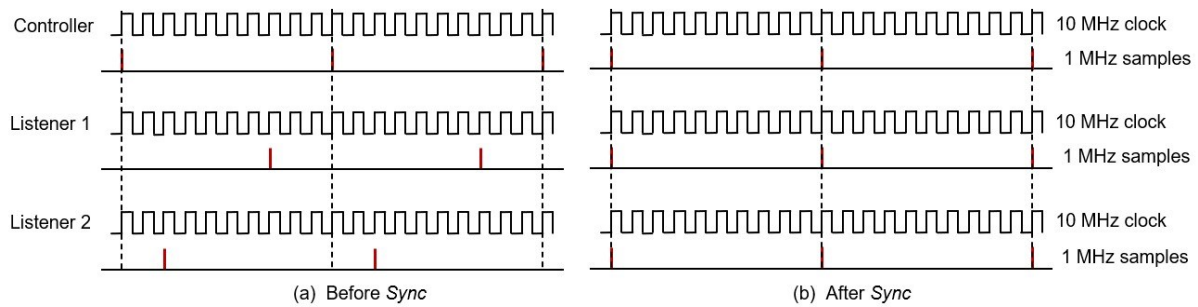


Figure 30. When multiple QDAC-II units are sharing a 10 MHz reference it is necessary to synchronize the voltage outputs by sending a trigger signal from one QDAC-II (the controller) to the other units (listeners) so that the 1 MHz sample output comes *in phase* across the units. After the synchronization pulse is sent from the controller, all units will output samples with inter delays less than 50 ns depending on the cabling. All instruments need to enter *Sync Wait* mode to react on the sync pulse.

Command	Description
SYSTem:CLOCk:SOURce	Defines the source of the main 10 MHz clock, either INTernal or EXTernal. In case of EXTernal clock, a 10 MHz square wave must be present on the Trigger In #4 (10MHz) BNC connector. The peak-to-peak amplitude should be at least 2.2 V and positive. Signals below 5 MHz are ignored.
SYSTem:CLOCk:EXTernal:STATus?	This command is used for reassuring that an external 10 MHz clock signal (on Trigger In #4) is being used. This is useful, because when setting SYST:CLOC:SOUR to EXT, the external signal is only used if the quality (amplitude) of the signal is acceptable.
SYSTem:CLOCk:SYNChronize[:IMMEDIATE]	This command (event) tells a listening unit to enter <i>Sync Wait</i> mode and listen for the next rising edge at <i>Sync In</i> (Trigger In #3) BNC connector. At that moment shift its 1 MHz sample clock to align with the next rising edge of the 10 MHz clock. The instrument will leave <i>Sync Wait</i> mode after 2 seconds if a signal has not been received.
SYSTem:CLOCk:SEND	Makes the instrument (the controller) output a 10 MHz clock on Trigger Out #5 ("10 MHz")
OUTPut:SYNChronize:SIGNal	Makes the instrument enter <i>Sync Wait</i> mode (see above) and thereafter sends a 10 $\mu$ s pulse out on the <i>Sync Out</i> (Trigger Out #4) BNC port. The signal will have positive polarity and will always be preceded by a 0V output of 10 $\mu$ s duration.

### Example – synchronizing using the master unit's clock

**Controller\*:**

```
>SYST:CLOC:SEND ON # Send 10 MHz clock out signal out on 10 MHz Out (Trigger Out #5).
```

**Listeners:**

```
>SYST:CLOC:SOUR EXT # Use external 10 MHz clock signal present on 10 MHz In (Trigger In #4).
>SYST:CLOC:SYNC # On next rising edge on Sync In (Trigger In #3) align 1 MHz sample.
```

**Controller – less than 2 seconds after**

```
>SYST:CLOC:SYNC # Enter Sync Wait mode and send pulse on Sync out (Trigger Out #4).
>OUTP:SYNC:SIGN
```

\*)If an external clock is used the Controller unit should have this command issued instead:

```
>SYST:CLOC:SOUR EXT           # Use external 10 MHz clock signal present on "10 MHz In" (Trigger In #4).
```

### Example – start the square wave generator simultaneously on synchronized units

```
ctrlr>OUTP:TRIG4:SOUR INT1     # Prepare Trigger Out #4 to react on Internal Trigger #1 on the controller.
ctrlr>SOUR4:SQU:TRIG:SOUR EXT4 # Make the square wave gen. on ch04 on the controller start on Ext. Trig. Input #4
ctrlr>SOUR4:SQU:INIT           # Initialise the square wave generator on ch04 on the controller

listen1>SOUR6:SQU:TRIG:SOUR EXT4 # Make the square wave gen. on ch06 on listening unit start on Ext. Trig. Input #4
listen1>SOUR6:SQU:INIT        # Initialise the square wave generator on ch06 on the listening unit #1

ctrlr>TINT 1                   # Fire Internal Trigger #1 on the controller. This will fire External trigger #4.
```



## 6 Operation

The QDAC-II command set adheres to the IEEE 488.2 (IEEE Standards Board, 1992) and SCPI standards (SCPI Consortium, 1999), with those adaptations required for exploiting the advanced functionalities of the instrument and omissions of those parts of the standards which do not contribute to instrument usability.

### 6.1 SCPI Command groups

In SCPI a command is represented by a *program header* consisting of one or more *instrument control headers*, also referred to as *program mnemonics*, or *keywords*, separated by colon characters “:”. So each command is a hierarchy of colon separated keywords; a command tree, where the keywords are branches and the colons are nodes. In SCPI terminology each branch are called a *sub-system*. Below is a list of the top level keywords – or *sub-systems* – for QDAC-II:

Command group (sub-system)	Description
*xxx	xxx represents one of the IEEE488.2 mandatory commands .
<b>ABORT</b>	Stops all triggered actions.
<b>SOURce</b>	Controls the voltage outputs.
<b>SENSe</b>	Controls the current sensors.
<b>READ</b>	Shortcut for read out of the current sensors.
<b>FETCH</b>	Reads measured current values.
<b>OUTPut</b>	Setup of Controls trigger outputs.
<b>TINT</b>	Direct control of internal triggers.
<b>TRACe</b>	Management for arbitrary waveform generation traces
<b>STATus</b>	Instrument status register readout and configuration.
<b>SYSTem</b>	Error read out, non-measurement related settings (e.g. LAN settings), synchronization.
<b>DIAGnostic</b>	Functionality diagnostics commands, calibration management etc.

## 6.2 SCPI Command syntax

Except for the IEEE 488.2 common command headers all starting with a “\*” character, most *instrument control header mnemonics* have a long form and a short form. The short form is the first four letters of the long form, except if the 4<sup>th</sup> letter is a vowel and the long form consists of more than 4 letters. Only the short form *or* the long form of command is accepted, not anything in between. However, short and long form mnemonics/keywords can be mixed in a program header. For example `sys:comm:lan:ipad?` for querying the instrument’s IP address can also be written as `system:comm:lan:ipaddress?`.

In the Command Reference (section 7), the short form of the mnemonics (keywords) are written in uppercase though the instrument does not distinguish between uppercase and lowercase characters. So uppercase and lower case can be mixed within a keyword.

Command lines should always end with a newline character <nl> (0x0A) – also known as line feed character. The instrument will accept carriage return <cr> (0x0D) character preceding the newline character.

*Command syntax*

`Level1:level2:level3 parameter1,parameter2,...`

*Query syntax*

`Level1:level2:level3? parameter1, parameter2,...`

### Compound command lines

A single command line can contain multiple program headers (commands) separated by a semicolon.

Keywords after a semicolon inherits the command tree up to the right most keyword of the first command, unless the keywords start with a colon, then it is starting at the root.

*Principle of the compound command syntax*

`cmd-1_Level1:cmd-1_level2:cmd-1_level3;cmd-2_level3;cmd-3_level3`

*Compound command syntax using colon (:) to reset the command tree*

`cmd-1_Level1:cmd-1_level2:cmd-1_level3::cmd-2_Level1:cmd-2_level2:cmd-2_level3`

### Example – setting the parameters and triggering a generator as a compound command

`>sour1:squ:freq 5000;span 0.2;dcyc 25;count 1000;trig:sour bus::*trg`

### Channel list syntax

Individual channels are addressed by adding a channel number suffix to the relevant keyword. Keywords which can have a channel number suffix:

Command/keyword
SOURce
SENSe
READ

FEtCh
STATus:OPERation:():CHANnel
STATus: QUEStionable:():CHANnel
DIAGnostics:VCALibration
DIAGnostics:ICALibration

Other commands which can have a numeric suffix:

Command/keyword
OUTPut:TRIGger
SYSTEM:TEMPerature

### Example – setting the dc (FIXed mode) output for a single channel

```
>sour10:volt 0.54
```

However, the instrument supports addressing multiple channels using the so-called channel list syntax. Instead of adding a numeric suffix to the relevant keyword a channel list may be added as the last parameter in a command. Channels lists can specify a list of individual channels or ranges of channels, or both in combination. Channel lists starts with “(“ and ends with “)“.

When the command is executed it is repeated for every channel in the list. Hence, the command is not executed in parallel for the channels but sequentially one channel at a time.

Channel list syntax	Description
(@1,3,5,6)	Comma separated. Addresses channels 1,3,5, and 6.
(@1:24)	Range, addresses all channels.
(@1:3,9,17)	Combined range and comma separated. Addresses channels 1,2,3,9, and 17.

### Example – setting all dc (FIXed mode) output to zero

```
>sour:volt 0,(@1:24)
```

Note that in the current version of the firmware (13-1.57) channels lists are not fully supported in compound messages. They only work for the last command in the compound program header.

## 6.3 Command synchronization

### 6.3.1 Sequential versus overlapped commands

There are two major categories of commands in a SCPI instrument as QDAC-II:

- **Sequential commands**
- **Overlapped commands**

**Sequential commands** are commands or functions which need to be fully executed before the next command can be executed. Hence, when sending a series of sequential commands to the instruments they will be executed in the order they are received one after the other. All commands except those mentioned below are sequential.

Note that sending a command is not the same as it will be executed immediately. When using LAN communication there can be up to 10 ms delay from a command is present at the LAN connector until it is executed, which should be taken into account.  
The safest way to ensure that the instrument has executed all sent commands is to perform a small query, for example \*STB?, and then wait for the response.

**Overlapped commands** are commands which execute while following commands are interpreted and executed. An overlapped command does not really have to be a command, it is in fact most often a trigger (internally or externally) which causes some action.

All triggered operations, and immediate setting of a DC voltage are overlapped commands. In order to check if a waveform generator, a DC:SWEep or DC:LIST is completed, their NCL (number of counts left) property can be checked. The same is true for the current sensors. When setting an immediate DC voltage one can test the LAST? property against a VOLT? query to see when the requested voltage has been reached. This may not be immediately due to finite slew rate; delays caused by the analogue low-pass filters are not accounted for.

### 6.3.2 Future section on Software synchronization (planned features)

### 6.3.3 Hardware Synchronization

The QDAC-II offers an advanced internal triggering system which is coupled to the outside world by the *Trig In* and *Trig Out* physical ports, see sections 5.4 and 5.5.

## 6.4 Status and events

In the current version of the firmware (13-1.57) only the status byte register (STB) has been implemented. Non implemented functionality is greyed out or is not described.

For monitoring the status of various operations and events, a SCPI instrument as the QDAC-II has a SCPI compliant status register structure, which is also used for reporting over current events. The registers are:

- Status Byte Register (STB).
- Standard Event Status Register (SESR).
- Operation Condition Register (OCR).
- Operation Event Register (OEVR).
- Questionable Condition Register (QCR).
- Questionable Event Register (QEVR).

The Status Byte Register (STB) and the Standard Event Status Register (SESR) are the two most important registers. They are both summary registers. The status byte (STB) contains summary information about occurred errors, the Error/Event Queue and the Output Queue. Bit 2 in the STB is raised when the Error queue is non-empty.

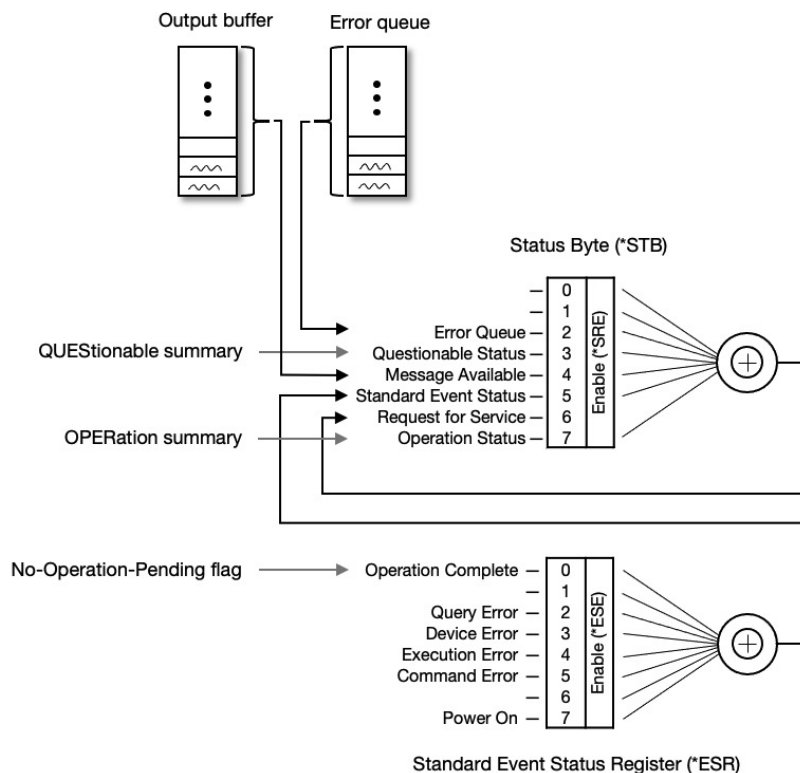


Table 2. STB (Status Byte Register).

Bit	Value	Name	Meaning
7	128	Operation Status	One or more of the enabled bits in the STATus:OPERation:EVENT register are set.
6	64	Request for Service	One or more of the service request enabled (*SRE) bits in the Status Byte Register have been set.
5	32	Standard Event Status	One or more enabled bits (*ESE) in the Standard Event Register are set.
4	16	Message Available	Data is available in the instrument's output buffer
3	8	Questionable status	One or more enabled bits in the Questionable Data Register are set (for QDAC-II, an overcurrent event has occurred).
2	4	Error Queue	One or more error messages are present in the Error Queue. These are read and cleared by the SYSTem:ERRor[:NEXT]? Command.
1	2	Not used	Value always 0.
0	1	Not used	Value always 0.

Future section: Table 3. Standard Event Status Register, ESR, Bits meaning.

#### 6.4.1 Future section: STATus:OPERation

#### 6.4.2 Future section: STATus:QUESTionable

## 6.5 Errors and events reporting

Error messages follow the scpi numbering with relevant device specific information added, giving relevant hints to the details of what has failed. A complete list of errors is given in Section 8.

For a general description of scpi errors see the SCPI Vol. 2 Command Reference (SCPI Consortium, 1999) section 21.8.

The error numbers fall in different categories and number ranges






Error number range	Group description
-499 .. -400	Query errors (standard)
-399 .. -300	Device errors (standard)
-299 .. -200	Execution errors (standard)
-199 .. -100	Command errors (standard)
300 .. 399	Device errors (specific)

Table 4 There are four standard SCPI error number (negative) ranges for different error categories. Positive error numbers are device specific.

### 6.5.1 Buzzer and LED

When an error occurs (including over current ILIMit), the buzzer will produce a beep (if not disabled), and the LED will start flashing a pattern indicating the type of error/event.

When an error or event is present the LED will only signal one error/event according to the priority shown in the table below (lowest number, highest priority). This means if both an error message is available and a current limit event has occurred, then it is the current limit event pattern which is signalled. Once the current limit event has been processed by the user, the LED will signal “SCPI error message available” until the error queue is empty, only interrupted if communication is taking place, or the instrument is restarted.

LED blink pattern (approx.)	State / error	Priority
	(Re)starting firmware	1
	Communicating	2
	Current limit event occurred	3
	SCPI error message available	4
	Waiting for command (idle)	5

## 7 Command Reference

### 7.1 IEEE Common commands

In the current version of the firmware (13-1.57) not all IEEE common commands are implemented, as the underlying status structures are not fully implemented.

#### ABORt

---

Description	Aborts all functions (functions generators, sweeps etc) and returns the trigger sequences to the idle / wait for trigger state.
Syntax	ABORt
Arguments	None
Query response	N/A
Example(s)	<p>&gt;SOUR:TRI:TRIG:SOUR BUS (@1:24) - sets the trigger source to BUS for triangle generators on all channels.</p> <p>&gt;*TRG - Fires the global BUS trigger starting the ch 1-24 triangle generators</p> <p>&gt; ABOR - Stops all triggered functions, including the 24 triangle generators above.</p>
Notes	This command will stop all waveform generators, DC:LIST or SWEeps, and current measurement. Note that if an output has not reached its set voltage due to slew rate limitations, then the output will continue ramping until the set value has been reached.

---

#### \*IDN?

---

Description	Identification query. Returns the product name, model number, serial number etc.
Syntax	*IDN?
Arguments	None
Query response	Returns the product name, model number, serial number, and firmware version.
Example(s)	<p>&gt;*IDN? QDevil,QDAC-II,Q310-1234,1.02</p>
Notes	

---

#### \*RST

---

Description	Reset command. Bring the QDAC into the power on state with all settings at their default values.
Syntax	*RST
Arguments	None
Query response	None. Event only.
Example(s)	>*RST

---



---

Notes This command does not restart the firmware. It merely aborts all ongoing jobs, sets output voltages to zero, and resets all data structures to their power up defaults and enters the “wait for message” idle state.

---

### \*TRG

---

Description Global trigger command. Triggers all functions which have BUS as their trigger source.

Syntax \*TRG

Arguments None

Query response N/A

Example(s) `> SOUR1:SQU:TRIG:SOUR BUS` - Sets the trigger source of the square wave generator on channel 1 to BUS  
`>*TRG` - Fires the global trigger starting the square wave generator on channel 1

Notes If no functions have their trigger source set to BUS, then the command has no effect.

---

### \*STB?

---

Description Status byte query. Returns the status byte register (STBR).

Syntax \*STB?

Arguments None

Query response Status byte (numeric response) in the range of 0-255, representing the sum of bits in the STB register, see Table 5.

Example(s) `>GARBage` - undefined command.  
`>*STB?`  
4

Notes

---

Table 5. STB (Status Byte Register) Bits meaning.

Bit	Value	Name	Meaning
7	128	Operation Status	One or more of the enabled bits in the Standard Operation Register are set.
6	64	Request for Service	One or more of the service request enabled (*SRE) bits in the Status Byte Register have been set.
5	32	Standard Event Status	One or more enabled bits (*ESE) in the Standard Event Register are set.
4	16	Message Available	Data is available in the instrument’s output buffer
3	8	Questionable status	One or more enabled bits in the Questionable Data Register are set.
2	4	Error Queue	One or more error messages are present in the Error Queue. These are read and cleared by the <code>SYSTEM:ERRor[:NEXT]?</code> Command.
1	2	Not used	Value always 0.
0	1	Not used	Value always 0.

## \*CLS

---

Description	Clear Status Command. Clears all status data structures and the error queue. This may also clear the status byte. Enable registers are not cleared.	
Syntax	*CLS	
Arguments	None	
Query response	N/A	
Example(s)	<pre>&gt;*STB? 4 &gt;*CLS &gt;*STB? 0</pre>	<ul style="list-style-type: none"><li>- Errors messages are available in the error queue</li><li>- All event registers are cleared</li><li>- The status byte register has been reset</li></ul>
Notes	<p>The data structures which are cleared are:</p> <ul style="list-style-type: none"><li>• <del>Standard Event Register (SERS)</del></li><li>• <del>OPERation Status Register (OSR)</del></li><li>• <del>QUESTionable Status Register (QSR)</del></li><li>• Error/Event Queue</li></ul> <p>As a consequence, the STB may also be cleared.</p> <p><a href="#">In the current firmware version (13-1.57) this command only clears the error/event queue</a></p>	

---

## 7.2 Future section: Interface commands

## 7.3 Voltage generators

### 7.3.1 Output modes

The QDAC has two voltage ranges. The purpose is to make best use of the 20 bits of resolution and to get best possible noise performance as the low voltage range exhibits lower noise than the high voltage range.

#### SOURce:VOLTage:RANGe

---

Description	Sets or reads the output voltage range for individual channels.	
Syntax	SOURce[n][:VOLTage]:RANGe[?] {LOW HIGH} [,channel_list]	
Arguments	LOW:	Low voltage range ( $\pm 2V$ ).
	HIGH:	High voltage range ( $\pm 10V$ ) (default).
Query response	Returns the current output voltage range for the selected channel(s).	
Example(s)	<pre>&gt;SOUR2:RANG LOW           - sets the voltage range for channel 2 to LOW. &gt;SOUR2:RANG?              - returns the current voltage range of channel 2 LOW</pre>	
Notes	If a voltage range of $\pm 2V$ is sufficient for your experiments the LOW range is recommended as this mode exhibits the lowest noise level and highest resolution.	

---

#### SOURce:VOLTage:RANGe:LOW

---

Description	Query only. Reads the calibrated MINimum or MAXimum output voltage for the LOW output voltage range for one or more channels.	
Syntax	SOURce[n][:VOLTage]:RANGe:LOW:{MINimum MAXimum}[?] [,channel_list]	
Arguments	MINimum:	Lowest calibrated voltage output.
	MAXimum:	Highest calibrated voltage output.
Query response	Returns the MINimum or MAXimum calibrated output voltage (in units of Volt) for the LOW voltage range of the selected channel(s).	
Example(s)	<pre>&gt;SOUR2:RANG:LOW:MIN?      - returns the minimum voltage for the LOW voltage range                            on chan. 2. -2.00346</pre>	
Notes	This command can for example be used by drivers to ensure that set commands are not setting voltages outside actual limits.	

---

#### SOURce:VOLTage:RANGe:HIGH

---

Description	Query only. Reads the calibrated MINimum or MAXimum output voltage for the LOW output voltage range for one or more channels.	
Syntax	SOURce[n][:VOLTage]:RANGe:HIGH:{MINimum MAXimum}[?] [,channel_list]	
Arguments	MINimum:	Lowest calibrated voltage output.
	MAXimum:	Highest calibrated voltage output.

---

---

Query response	Returns the MINimum or MAXimum calibrated output voltage (in units of Volt) for the HIGH voltage range of the selected channel(s).
Example(s)	<pre>&gt;SOUR2:RANG:HIGH:MAX? - returns the maximum voltage for the HIGH voltage range on chan. 2.</pre> <pre>10.00321</pre>
Notes	This command can for example be used by drivers to ensure that set commands are not setting voltages outside actual limits.

---

The QDAC-II has three low pass filter options, DC, MEDium and HIGH, for use in different experimental situations: The lower the bandwidth, the lower the noise level. They all two-pole filters. For outputting fast waveforms the HIGH filter option should be used. For DC, slow sweeps and slow waveforms MEDium should be used. The DC filter cut-off is around 10 Hz, so this mode is only use for changing voltages very slowly, or for keeping them at a DC level for a long time. 25 bits of resolution is default in this mode except when executing LISTs or SWEeps.

### SOURce:VOLTage:FILTer:LOWPass

---

Description	Sets or reads the output lowpass filter frequency for individual channels.	
Syntax	SOURce[n]::VOLTage]:FILTer[:LOWPass][?] {DC MEDium HIGH} [,channel_list]	
Arguments	DC:	Cut-off $\approx$ 10 Hz, use for keeping voltage es stable and precise.
	MEDium:	Cut-off $\approx$ 10kHz, use for DC, slow ramps and slow waveforms.
	HIGH:	Cut-off $\approx$ 100kHz, use for fast ramps and fast waveforms.
Query response	Returns the current filter cut-off setting the selected channel(s).	
Example(s)	<pre>&gt;SOUR2:FILT DC - sets the lowpass filter for chan. 2 to DC.</pre> <pre>&gt;SOUR2:FILT? - returns the current lowpass filter setting for chan. 2</pre> <pre>DC</pre>	
Notes		

---

## 7.3.2 DC generator

Not that changing any parameter SWEep parameter while SWEep is running or any LIST parameter while LIST is running, will end their current trigger cycle and thereby stop the sequence and bring the DC generator trigger state back to the *Initiated* state if it is in INITiate:CONTinuous mode or otherwise to is *Idle* state.

### SOURce:DC:VOLTage:MODE

---

Description	Sets or reads the DC generator mode, and hence determines which commands controls the Amplitude sub-system. In FIXed mode (constant poutput voltage) the voltage is controlled by LEVel, in SWEep mode the voltage is controlled by START and STOP, and in LIST mode the sequence of voltages is given by the LIST:TVOLTage subsystem. When SLEWrate is <> INF the voltage will ramp from value to value at a finite rate.	
Syntax	SOURce[n]::DC]::VOLTage]:MODE[?] FIXed SWEep LIST [,channel_list]	

---

---

Arguments	<p>FIXed: Constant output (default).</p> <p>SWEep: Outputs a voltage sweep according to the SWEep subsystem settings.</p> <p>LIST: Outputs a voltage sequence according to the LIST subsystem settings.</p>
Query response	Returns the current DC generator MODE.
Example(s)	<p>&gt;SOUR2:DC:MODE FIX - sets the DC generator to FIXed mode channel 2.</p> <p>&gt;SOUR2:DC:MODE? - returns the current MODE of the DC generator of ch. 2</p> <p>FIX</p>
Notes	The IMMEDIATE and TRIGGER(ed) values in the FIXed mode are always updated with the most recent output value from the SWEep and LIST systems. Therefore, there will no voltage jumps when switching from SWEep or LIST to FIXed or vice versa. However, as soon as either a SWEep or LIST sequence are triggered there may be a jump to the start value (limited by SLEWrate and analogue filters).

---

### SOURce:DC:VOLTage:LEVel:IMMediate:AMPLitude

---

Description	<p>Sets or reads the DC generator voltage component of the output of a selected channel(s) in FIXed mode.</p> <p>When SLEWrate is not INF the voltage will ramp to the set value at a finite rate. Note that this set value is always updated whenever a new DC value is set by the Voltage:TRIGGER, LIST, or SWEep sub-systems.</p>
Syntax	SOURce[n]:DC:VOLTage[:LEVel[:IMMediate[:AMPLitude]]][?]<numeric_value> [channel_list]
Arguments	<p>&lt;numeric_value&gt;: DC voltage in Volts or MINimum or MAXimum</p> <p>Minimum value : -10 or -2V, depending on the range, and calibrated min/max</p> <p>Maximum value: +10 or +2V, depending on the range, and calibrated min/max</p> <p>Default: 0</p>
Query response	<p>Returns the current DC component of the channel's output. If reading before the set value has been reached (due to finite slewrate), the read value may differ from the set value. To read out the <u>set</u> value, use</p> <p>SOURce[n]:DC:VOLTage[:LEVel[:IMMediate[:AMPLitude]]]:LAST? [chanlist]</p>
Example(s)	<p>&gt;SOUR2:VOLT 1.12 - sets the DC level to 1.12 for channel 2 immediately.</p> <p>&gt;SOUR2:VOLT? - returns the actual temporal DC voltage component of ch.1 2</p> <p>1.12</p>
Notes	<p>If the DC generator is not in FIXed mode this command has no effect for setting a voltage, only for reading.</p> <p>If SOURce:FILTer = DC and RENHancemen= ON (default) the resolution is enhanced to 25 bit. The resolution enhancement can be disabled by the SOURce:DC:RENHancement command.</p>

---

### SOURce:DC:VOLTage:LEVel:IMMediate:AMPLitude:LAST?

---

Description	Reads the most recent DC voltage set by the IMMEDIATE command, or any of the methods overwriting it, on the channel(s) in FIXEd mode. This is a query only command making it possible to read the last set value which level might not yet have been reached.
Syntax	SOURce[n]:[DC]:VOLTage[:LEVel[:IMMediate[:AMPLitude]]]:LAST? [,channel_list]
Arguments	none
Query response	Returns the most recently set DC component of the channel's output. The set level might not have been reached at the time of query if a finite slew rate is used.
Example(s)	<code>&gt;SOUR2:VOLT:LAST?</code> - returns the last set DC voltage on channel 2 1.5
Notes	<a href="#">In the current version of the firmware (13-1.57) the LAST command does not work as designed</a>

---

### SOURce:DC:VOLTage:LEVel:TRIGger:AMPLitude

---

Description	Sets or reads the next triggered DC voltage output of a selected channel for FIXEd mode. Note that the value is always updated whenever a new DC value is set by the VOLTage:IMMediate sub system or when changed by the LIST, or SWEep sub-systems so that a spurious trigger will not change the output unintentionally.
Syntax	SOURce[n]:[DC]:VOLTage[:LEVel]:TRIGger[:AMPLitude][?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: DC voltage in Volts or MINimum or MAXimum Minimum value: -10 or -2V, depending on the range, and calibrated min/max Maximum value: +10 or +2V, depending on the range, and calibrated min/max Default: 0
Query response	Returns the value set by the most recent TRIG command or as the value always follow the IMMEDIATE, SWEep and LIST subsystems, the most recent value set by any of them.
Example(s)	<code>&gt;SOUR2:VOLT:TRIG 3</code> - sets the next triggered DC level to 3 for channel 2. <code>&gt;SOUR2:VOLT:TRIG?</code> - returns the next triggered DC voltage output of ch.1 3
Notes	When SLEWrate is not INF the voltage will ramp to the set value at a finite rate when the trigger event occurs. See also notes for SOURce:DC:VOLTage:LEVel:IMMediate:AMPLitude. <a href="#">In the current version of the firmware (13-1.57) the TRIG value is unfortunately propagated (triggered) when switching the filter for the channel to or from DC. So please avoid leaving an untriggered value in this register. (In fact it will happen if RENHancement is ON, which it is by default, but not if is off).</a>

---

### SOURce:DC:VOLTage:SLEW

---

Description	Sets or reads the DC generator slew rate of a channel. When a finite slew rate is set, the rate at which the voltage can change is limited and transients from abruptly stepping the voltage are avoided. The slew rate is active for all DC modes (FIXEd, LIST, SWEep).
-------------	---

---

Syntax	SOURce[n][:DC]:VOLTage:SLEW[?] <numeric_value> [,chlist]	
Arguments	<numeric_value>:	Maximum voltage change rate in Volts per second (V/s)
	Minimum value :	0.01
	Maximum value:	2e7 corresponds to - 20V/1 $\mu$ s (same as INF)
	Default:	INF
Query response	Returns the current DC generator slew rate of the specified channel(s).	
Example(s)	>SOUR2:VOLT:SLEW 200	- sets the DC slew rate for channel 2 to 200 V/s.
	>SOUR2:VOLT:SLEW?	- returns set DC slew rate for channel 2
	200	
Notes	<p>Depending on the magnitude of voltage steps, the set slew rate may or may not have an effect on the generated signal. If for example the difference in voltage between subsequent values divided by the sampling interval of 1<math>\mu</math>s is smaller than the slew rate, the slew rate will have no effect. Also, if the voltage difference is for example just one DAC increment, the only observed effect would be a delay in going from the present value to the next.</p> <p>Ultimately the slew rate is limited by the hardware, i.e. by the choice of low pass filter and current measurement range.</p> <p><i>A small transient (spike) must be expected to appear when changing the slew rate at nonzero voltage outputs.</i></p>	

### SOURce:DC:SWEep:TIME?

Description	Reads the DC generator sweep time, i.e. the duration of a single ramp.	
Syntax	SOURce[n][:DC]:SWEep:TIME[?] [chlist]	
Arguments	none	
Query response	Returns the current DC generator SWEep TIME of the specified channel(s).	
Example(s)	>SOUR2:SWE:TIME?	- returns the SWEep time for channel 2
	0.05	
Notes		

### SOURce:DC:SWEep:DWELI

Description	Sets or reads the DC generator sweep dwell time, i.e. the duration of each level of a stepped ramp.	
Syntax	SOURce[n][:DC]:SWEep:DWELI[?] <time> [,chlist]	
Arguments	<time>:	Ramp/sweep time in seconds
	Minimum value :	2e-6 (1 sample)
	Maximum value:	36000 (10 hours)
	Default:	2e-6
Query response	Returns the current DC generator SWEep DWELI time of the specified channel(s).	

Example(s) `>SOUR2:SWE:DWEL 0.01` - sets the sweep dwell time for channel 2 to 10ms.  
`>SOUR2:SWE:DWEL?` - returns the sweep dwell time for channel 2  
`0.01`

Notes If DWEL: AUTO is ON, the query of DWEL: TIME will return the auto calculated value. Note that DWEL is simply the time between consecutive voltage steps. It is up to the user to choose DWEL times high enough for reaching a stable voltage level between voltage steps. How long it should be depends on the chosen OUTPut: FILTer, and the SLEWrate. IF DWELL is auto calculated, it may be less than the minimum value. As a step length cannot be less than a two sample periods (2  $\mu$ s) the resulting sweep will have fewer steps than POINTs and the step lengths (their dwell time) will be uneven. Step lengths will also be uneven if DWELL is not an integer number of sampling intervals.

### SOURce:DC:SWEep:POINTs

Description Sets or reads the number of points in a stepped sweep. This parameter is not used if GENERation is set to ANALog. POINTs is coupled to STEP and SPAN as:  
 $POINTs = SPAN / STEP + 1$   
 If the value of STEP is changed POINTs will automatically be changed, but not when SPAN is changed.  
 POINTs is also coupled to TIME

Syntax `SOURce[n]:[DC]:SWEep:POINTs[?] <numeric_value> [,chlist]`

Arguments `<numeric_value>`: Number of points  
 Minimum value : 1  
 Maximum value: 65536  
 Default value: 100

Query response Returns the number of points for stepped sweeps of the channel(s).

Example(s) `>SOUR2:SWE:POIN 101` - sets the number of sweep points for ch. 2 to 101. There will be 100 levels.  
`>SOUR2:SWE:POIN?` - returns the DWEL: AUTO state for channel 2  
`101`

Notes A sweep with only one point will thus just set the initial (START) voltage and then ends after one DWEL time. For ANALog sweeps, it is easiest to set POINTs = 1 and then let DWEL define the sweep TIME (in multiples of the sample rate which is 1e-6 seconds)

### SOURce:DC:SWEep:GENeration

Description GENERation defines whether the sweep is stepped or smooth (analog). In case the sweep is ANALog, the parameters POINTs and STEP are ignored.

Syntax `SOURce[n]:[DC]:SWEep:GENeration [?] STEPped|ANALog [,chlist]`

Arguments STEPped: The sweep will be a stair case sweep  
 ANALog: The sweep will be a linear ramp, only discretized by the sampling rate and the DAC precision

Query response Returns the current DC generator sweep generation type.



---

Example(s) `>SOUR2:SWE:GEN ANAL` - sets the SWEep type to be a linear ramp for channel 2.  
`>SOUR2:SWE:GEN?` - returns the current SWEep type for channel .2  
`ANAL`

Notes The duration (TIME) for an ANALog SWEep is given by DWELI x POINTs, even though these two quantities have no individual meaning for an ANALog SWEep. For ANALog sweeps, it is easiest to set POINTs = 1 and then define the sweep TIME by setting DWELI to the requested sweep time in multiples of the sample rate, which is 1e-6 seconds.

---

### SOURce:DC:SWEep:COUNT

---

Description Defines the number of repetitions of the definedsweep.

Syntax `SOURce[n][:DC]:SWEep:COUNT[?] <numeric_value> [,channel_list]`

Arguments `<numeric_value>`: integer, number of repetitions.  
 Minimum value : 0  
 Maximum value:  $2^{24} - 1$  (16777215), or INF  
 Default: 1

Query response Returns the number of periods/cycles produced when the sweep is started.

Example(s) `>SOUR2:SWE:COUN 5` - sets the number of repetitions to 5 for the sweep  
`>SOUR2:SWE:COUN?` - returns the number of repetitions  
`5`

Notes Note that SWEep and LIST are not intended to run indefinitely but can when COUNT is set to INF.

---

### SOURce:DC:SWEep:NCLeft

---

Description Queries the number of remaining cycles (repetitions) of a running SWEep.

Syntax `SOURce[n][:DC]:SWEep:NCLeft? [,channel_list]`

Arguments none

Query response Returns the number of remaining repetitions of the sweep including an ongoing sweep, when the sweep has been triggered. Right after triggering NCLeft will return COUNT. During the last iteration it will return 1. If the generator is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned.

Example(s) `>SOUR2:SWE:NCL?` - returns the number of remaining repetitions of the DC sweep on ch 2  
`47`

Notes This command is primarily intended for making it possible for a driver to re-establish the instrument state approximately, if it becomes disconnected to the instrument. But it can also serve as a means for finding out if a sweep has completed.

---

### SOURce:DC:SWEep:VOLTage:STARt

---

Description	Command for setting or reading the initial voltage for a sweep.	
Syntax	SOURce[n]:[DC]:SWEep[:VOLTAGE]:STARt[?] <numeric_value> [,chlist]	
Arguments	<numeric_value>:	DC voltage in Volts or MINimum or MAXimum
	Minimum value :	-10 or -2V, depending on the range, and calibrated min/max
	Maximum value:	+10 or +2V, depending on the range, and calibrated min/max
	Default:	0
Query response	Returns the current DC generator SWEep start (initial) voltage.	
Example(s)	>SOUR2:SWE:STAR 1.1	- sets the SWEep start voltage to 1.1V on ch. 2.
	>SOUR2:SWE:STAR?	- returns the current SWEep start voltage for ch. 2.
	1.1	
Notes		

---

### SOURce:DC:SWEep:VOLTage:STOP

---

Description	Command for setting or reading the final voltage for a sweep.	
Syntax	SOURce[n]:[DC]:SWEep[:VOLTage]:STOP[?] <numeric_value> [,chlist]	
Arguments	<numeric_value>:	DC voltage in Volts or MINimum or MAXimum
	Minimum value :	-10 or -2 (Volt), depending on the range, and calibrated min/max
	Maximum value:	+10 or +2 (Volt), depending on the range, and calibrated min/max
	Default:	0
Query response	Returns the current DC generator SWEep stop (final) voltage.	
Example(s)	>SOUR2:SWE:STOP 2.3	- sets the SWEep end voltage on ch. 2 to 2.3V.
	>SOUR2:SWE:STOP?	- returns the current SWEep end voltage for ch. 2.
	2.3	
Notes		

---

### SOURce:DC:LIST:COUNT

---

Description	Defines the number of repetitions carried out for each triggering of LIST execution.	
Syntax	SOURce[n]:[DC]:LIST:COUNT[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>:	integer, number of repetitions.
	MINimum value :	0
	MAXimum value:	$2^{24} - 1$ (16777215), or INF
	Default:	1
Query response	Returns the number of periods/cycles produced when the LIST sequence is started.	
Example(s)	>SOUR2:LIST:COUN 5	- sets the number of repetitions to 5 for the list sequence
	>SOUR2:LIST:COUN?	- returns the number of repetitions
	5	

---

Notes Note that SWEEP and LIST are not intended to run indefinitely but can when COUNT is set to INF.

### SOURce:DC:LIST:NCLeft

Description	Queries the number of remaining repetitions of a running LIST sequence	
Syntax	SOURce[n][:DC]:LIST:NCLeft? [,channel_list]	
Arguments	None	
Query response	Returns the number of remaining repetitions of the LIST sequence (not the number of remaining steps in the sequence) including the ongoing iteration, when the LIST has been triggered. Right after triggering NCLeft will return COUNT. During the last iteration a value of 1 will be returned. If the generator is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned.	
Example(s)	>SOUR2:LIST:NCL?  6	- returns the number of remaining repetitions of the LIST sequence on ch 2
Notes	This command is primarily intended for making it possible for a driver to re-establish the instrument state approximately, if it becomes disconnected to the instrument. But it can also serve as a means for finding out if a LIST sequence has completed.	

### SOURce:DC:LIST:VOLTage

Description	Command for setting or reading the voltages for a LIST sequence.	
Syntax	SOURce[n][:DC]:LIST:VOLTage {<binary block> <list of numeric_values>} [,chlist] SOURce[n][:DC]:LIST:VOLTage? [chlist]	
Arguments	<list of numeric_values>: Comma separated list of voltages in units of Volts. <binary block>: IEEE 488.2 binary block (floating point values in Volts). Minimum value : -10 or -2V, depending on the range. Maximum value: +10 or +2V, depending on the range. Default: 0	
Query response	Returns the current DC generator LIST voltage sequence in the format specified by the FORMat subsystem.	
Example(s)	>SOUR1:LIST:VOLT 0, 1, 2, 3 >SOUR2:LIST:VOLT #18*****  >SOUR2:LIST:VOLT? 0.345, 1.982	- sets the LIST sequence to 0,1,2, and 3V on ch. 1. - sets the LIST sequence to the two 4 byte floating point numbers symbolized by the asterisks on ch. 2. - returns the current LIST sequence voltages of ch. .

---

**Notes** The maximum number of values which can be transferred in the comma separated format is 1024. Use the :APPend command to extend the list – or use the binary format for large lists.  
The maximum length of a LIST is 65536 points.  
A value error is produced if any of the voltages exceeds the voltage limits of the channel(s) in the currently set RANGE(s).  
Clipping will occur when executing the LIST if the produced voltages are outside the limits when the channel is in the LOW RANGE mode. (Voltages may have been set while in the HIGH RANGE mode).

---

### SOURce:DC:LIST:VOLTage:APPend

**Description** Command for adding voltages to a LIST sequence.

**Syntax** SOURce[n]:[DC]:LIST:VOLTage:APPend {<binary block>|<list of numeric\_values>} [chlist]

**Arguments**

<list of numeric_values>:	Comma separated list of voltages in units of Volts.
<binary block>:	IEEE 488.2 binary block (floating point values in Volts)
Minimum value :	-10 or -2V, depending on the range, and calibrated min/max.
Maximum value:	+10 or +2V, depending on the range, and calibrated min/max.
Default:	0

**Query response** None

**Example(s)** `>SOUR2:LIST:VOLT:APP 5, 4, 3` - adds voltages 5, 4, 3 to the ch. 2 LIST sequence.

**Notes** The maximum number of values which can be transferred in the comma separated format at a time is 1023. For more points please use this command repeatedly or use the binary format for large lists (maximum 65536 points in total).  
A value error is produced if any of the voltages exceeds the voltage limits of the channel(s) in the currently set RANGE(s).  
Clipping will occur when executing the LIST if the produced voltages are outside the limits when the channel is in the LOW RANGE mode. (Voltages may have been set while in the HIGH RANGE mode).

### SOURce:DC:LIST:DWELI

---

**Description** Sets or reads the DC LIST generator dwell time, i.e. the duration of each voltage in a LIST sequence

**Syntax** SOURce[n]:[DC]:LIST:DWELI[?] <numeric\_value> [chlist]

**Arguments**

<numeric_value>:	Time spent at each voltage level in the LIST sequence.
Minimum value :	2e-6 (1 sample)
Maximum value:	36000 (10 hours)
Default:	1e-3

**Query response** Returns the current DC generator LIST DWELI time of the specified channel(s).

**Example(s)** `>SOUR2:LIST:DWEL 0.01` - sets the list dwell time for channel 2 to 10ms.  
`>SOUR2:LIST:DWEL?` - returns the list dwell time for channel 2  
`0.01`

---

---

**Notes** Note that DWELL is simply the time between consecutive steps in voltage. It is up to the user to choose DWELL times high enough for reaching a stable voltage level between voltage steps. How long it should be depends on the chosen OUTPUT:FILTER, and the SLEWrate.

---

### SOURCE:DC:LIST:DIRrection

---

**Description** Specifies the direction that the sequence list is scanned. If UP is selected, the list is scanned from the first to the last item in the sequence list. If DOWN is selected, the list is scanned from the last item to the first item in the sequence list

**Syntax** SOURCE[n][:DC]:LIST:DIRrection[?] UP|DOWN [,chlist]

**Arguments** UP: LIST execution order goes from first element to last element (default).  
DOWN: LIST execution order goes from last element to first element.

**Query response** Returns the current DC LIST generator list sequence direction.

**Example(s)** >SOURCE2:LIST:DIR DOWN - sets the LIST direction to down for channel 2.  
>SOURCE2:LIST:DIR? - returns the current LIST direction for channel .2  
DOWN

**Notes**

---

### SOURCE:DC:LIST:POINTS?

---

**Description** Queries the number of points in a LIST sequence. Query only command.

**Syntax** SOURCE[n][:DC]:LIST:POINTS? [,chlist]

**Arguments** None

**Query response** Returns the number of points in the LIST of one or more channels.

**Example(s)** >SOURCE2:LIST:POIN? - returns the number of points in the LIST for channel 2  
4

**Notes**

---

### SOURCE:DC:LIST:TMODe

---

**Description** Specifies the trigger mode for the LIST sequence, whether triggering should start the automatic run of the sequence or should just advance one point.

**Syntax** SOURCE[n][:DC]:LIST:TMODe? {AUTO|STEPped} [,chlist]

**Arguments** AUTO: (Default) Specifies that a trigger event should start the automatic run of the LIST sequence, advancing points automatically.  
STEPped: Every time the LIST sequence receives a trigger signal it advances to the next value in the list.

**Query response** Returns the current trigger mode setting.

---

---

Example(s)	<pre>&gt;SOUR2:LIST:TMODE? AUTO &gt;SOUR2:LIST:TMODE STEP</pre>	<p>- Returns the current trigger mode setting for the LIST sequence on channel 2.</p> <p>- Sets the trigger mode for the LIST sequence on channel 2 to STEPped.</p>
Notes	<p>In both modes the sequence has to be in the INITiated state in order to respond to trigger signals.</p>	

---

### SOURce:DC:DAC:LEVel:IMMediate:AMPLitude

---

Description	<p>Sets or reads the binary DAC value representing the DC voltage component of the output of a selected channel(s) in FIXed mode. When SLEWrate is not INF the voltage will ramp to the set value at a finite rate. Note that this <i>set</i> value is always updated whenever a new DC value is set by the Voltage:TRIGger, LIST, or SWEep sub-systems.</p>	
Syntax	<pre>SOURce[n][:DC]:DAC[:LEVel[:IMMediate[:AMPLitude]]][?] &lt;numeric_value&gt; [,channel_list]</pre>	
Arguments	<pre>&lt;numeric_value&gt;: Integer representing the 20-bit DAC digital value. MINimum value : -524288 MAXimum value: 524287 Default:        0</pre>	
Query response	<p>Returns the current DC component of the channel's output. If reading before the set value has been reached (due to finite slewrate), the read value may differ from the set value.</p>	
Example(s)	<pre>&gt;SOUR2:DAC 22040 - sets the DC level to digital 22040 for channel 2 immediately. &gt;SOUR2:DAC?      - returns the actual temporal DC voltage component of channel                   2 in DAC binary unconverted digital units  22040</pre>	
Notes	<p>This command is primarily used during calibration of the voltage outputs of the instrument. Therefore, it is important to stop (ABORt) all waveform generators in this mode. Otherwise the result may be unexpected. If the DC generator is <u>not</u> in FIXed mode this command has no effect. As only 20 bit DAC values can be specified it makes no the different if resolution enhancement is active or not.</p>	

---

### SOURce:DC:RENHancement

---

Description	<p>Enables or disables resolution DC mode (low bandwidth). When enabled the resolution in FILTer:DC mode will be 25 bit.</p>	
Syntax	<pre>SOURce[n][:DC]:RENHancement][?] {ON OFF} [,channel_list]</pre>	
Arguments	<pre>ON :           25 bit resolution in DC mode (20 bits in MEDium and HIGH filter                 modes) OFF:           20 bit resolution in DC mode Default:      ON</pre>	
Query response	<p>Returns the active state of resolution enhancement of the specified channel(s).</p>	

---

---

Example(s) `>SOUR2:RENH OFF` - Disables resolution enhancement on ch. 2.  
`>SOUR2:RENH?` - Returns the current state (ON/OFF) of resolution enhancement on ch. 2.  
`OFF`

Notes 25 bits resolution is by default enabled for DC (low bandwidth) mode but can be switched off.  
The setting is reset upon power cycling or when executing `*RST` or `SYSTEM:RESet`.

---

### 7.3.3 Sine wave generator

Note that changing any parameter, while the generator is running, will end its current trigger cycle and thereby stop the generator and bring its trigger state back to the *Initiated* state if it is in INITiate:CONTInuous mode or otherwise to is *Idle* state. To re-run, it has to be re-triggered (CONTInuous mode) or re-initiated and re-triggered.

#### SOURce:SINE:PERiod

---

Description	Defines the period for the sine wave function generator. Overrides a previously set FREQUENCY.
Syntax	SOURce[n]:SINE:PERiod[?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: Specifies the period in seconds. Minimum value (s): 2e-6 Maximum value (s): 3600 Default (s): 0.001
Query response	Returns the period (in seconds) for the queried generator or a list of periods in case multiple channels are queried. The reported period will be the same as the value set by the user and not the actual outputted period in case it has been rounded (see notes).
Example(s)	<pre>&gt;SOUR2:SINE:PER 0.0001 - sets the sine wave period on BNC2 to 0.1ms (10kHz). &gt;SOUR:SINE:PER 0.0001, (@1:24) - sets the sine wave period on BNC1-24 to 0.1ms. &gt;SOUR2:SINE:PER? - returns the period of the sine wave generator on BNC2. 0.0001 &gt;SOUR:SINE:PER? (@2,3,10) - returns the sine wave period of BNCs 2,3 and 10. 0.0001, 0.0001, 0.00002</pre>
Notes	From a period of around 65 ms and above, the sine is not calculated for every 1µs sample, but will be interpolated. Note that not all periods, especially small periods, will give a symmetric waveform. Please see the block about <i>Waveform distortion and magic</i> periods and duty-cycles in section 5.2.2. Further, the generated curve is smoothed by the low-pass filters (section 5.1.2).

---



## SOURce:SINE:FREQuency

---

Description	Defines the frequency for the sine wave function generator. Overrides a previously set PERiod.	
Syntax	SOURce[n]:SINE:FREQuency[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>: Specifies the frequency in Hz. MINimum value (Hz): 0.00027778 (corresponding to a 1 hour period) MAXimum value (Hz): 5e5 Default (Hz): 1000	
Query response	Returns the frequency (in Hz) for the queried generator or a list of frequencies in case multiple channels are queried. The reported frequency will be the same as the value set by the user and not the actual outputted frequency in case the corresponding PERiod has been rounded (see notes).	
Example(s)	<pre>&gt;SOUR2:SINE:FREQ 25000</pre> - sets the sine wave frequency on BNC2 to 25kHz. <pre>&gt;SOUR2:SINE:FREQ?</pre> - returns the frequency of the sine wave generator on BNC2. <pre>25000</pre>	
Notes	From around 15Hz and below, the sine is not calculated for every 1µs sample, but will be interpolated. See note on magic periods giving a faithful reproduction of the waveform under :PERiod.	

---

## SOURce:SINE:COUNT

---

Description	Defines the number of periods which are generated for one trigger.	
Syntax	SOURce[n]:SINE:COUNT[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>: integer, number of repetitions. -1 denotes infinite INFinite: Makes the generator run indefinitely Minimum value : -1 same as INFinite Maximum value: $2^{24} - 1$ (16777215) Default: -1	
Query response	Returns the number of periods/cycles produced when the generator is started.	
Example(s)	<pre>&gt;SOUR2:SINE:COUN 5</pre> - sets the number of repetitions to 5 for the sine wave <pre>&gt;SOUR2:SINE:COUN?</pre> - returns the number of repetitions <pre>5</pre>	
Notes		

---

## SOURce:SINE:NCLeft

---

Description	Queries the number of remaining cycles which are generated for one trigger.	
Syntax	SOURce[n]:SINE:NCLeft? [,channel_list]	
Arguments	none	

---

**Query response** Returns the number of remaining periods/cycles produced of a running generator, including the current cycle. Right after triggering NCLeft will return COUNT and will return 1 during the last cycle.  
If the generator is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned.  
After being triggered, if COUNT is INF (-1) then -1 is returned.

**Example(s)** `>SOUR2:SINE:NCL?` - returns the number of remaining repetitions for the sine wave generator on ch 2  
3

**Notes** This command is primarily intended for making it possible for a driver to re-establish the instrument state approximately, if it becomes disconnected to the instrument.  
But it can also serve as a means for finding out if the generator has finished running.

### SOURce:SINE:POLarity

**Description** Defines the if the positive part of a cycle comes before (normal) or after the negative part.

**Syntax** `SOURce[n]:SINE:POLarity[?] {NORMAL|INVERTed} [,channel_list]`

**Argument** `NORMAL` default, the positive signal comes before the negative part  
`INVERTed:` the negative portioin of a period before the positive portion

**Query response** Returns the polarity of the sine wave generator.

**Example(s)** `>SOUR2:SINE:POL INV` - *inverts the polarity of the sine wave generator*  
`>SOUR2:SINE:POL?` - returns the polarity  
`INV`

**Notes**

### SOURce:SINE:VOLTage:SPAN

**Description** Defines the peak to peak voltage span of the generated sine wave, with the centre being equal to the OFFset (default zero).

**Syntax** `SOURce[n]:SINE[:VOLTage]:SPAN [?] <numeric_value> [,channel_list]`

**Arguments** `<numeric_value>:` peak to peak voltage, or MINimum or MAXimum  
Minimum value : 0V  
Maximum value: +20V or +4V, depending on the present range  
Default: 0.2 V

**Query response** Returns the top to bottom voltage of the square generator.

**Example(s)** `>SOUR2:SIN:SPAN 2` - *sets peak to peak amplitude to 2V*  
`>SOUR2:SIN:SPAN?` - returns the peak to peak amplitude  
2

**Notes** If the SPAN value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting signal is within the actual voltage range.  
The signal will simply be clipped if it is.

### SOURce:SINE:VOLTage:OFFSet

---

Description	Defines the relative offset of the sine wave generator signal. If OFFSet = 0 the sine wave signal will just swing symmetrically around the source's DC value (including the sum of offsets of any other generators).
Syntax	SOURce[n]:SINE[:VOLTage]:OFFSet [?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: offset in volts Minimum value : -10 or -2V, depending on the range (plus half the SPAN) Maximum value: +10 or +2V, depending on the range (minus half the SPAN) Default: 0
Query response	Returns the added offset to the center position of the sine wave signal
Example(s)	>SOUR2:SINE:OFFS 0.7 - sets the offset to +0.7V for the sq. wave generator on ch 2. >SOUR2:SINE:OFFS? - returns the relative offset 0.7
Notes	The OFFset is intended for compensation and not as an alternative way of setting a DC value, as it is only applied when the generator is running. If the OFFSet value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting signal is within the actual voltage range. The signal will simply be clipped if it is.

---

## SOURce:SINE:VOLTage:SLEW

---

Description	Sets or reads the sine wave generator slew rate of a channel. This is mostly relevant for the initial offset applied when the generator starts (and when it ends), to avoid jumps. If the SLEW rate is lower than the steepest part of the sine curve, the sine will become distorted and/or a phase lag will be present.
Syntax	SOURce[n]:SINE[:VOLTage]:SLEW[?] <numeric_value> [,chlist]
Arguments	<numeric_value>: Maximum voltage change rate in Volts per second (V/s) Minimum value : 0.01 Maximum value: 2e7 - 20 V/1 $\mu$ s (same as INF) Default: INF
Query response	Returns the current sine wave generator slew rate of the specified channel(s).
Example(s)	>SOUR2:SINE:VOLT:SLEW 200 - sets sine wave slew rate for channel 2 to 200 V/s. >SOUR2:SINE:SLEW? - returns the set sine wave slew rate for channel 2 200
Notes	Depending on the amplitude of the waveform, the set slew rate may or may not have an effect on the generated signal. Ultimately the slew rate is limited by the hardware, i.e. by the choice of low pass filter and current measurement range. <i>A small transient (spike) must be expected to appear when changing the slew rate at non zero voltage outputs.</i>

---

### 7.3.4 Square wave generator

Not that changing any parameter, while the generator is running, will end its current trigger cycle and thereby stop the generator and bring its trigger state back to the *Initiated* state if it is in INITiate:CONTinuous mode or otherwise to is *Idle* state. To re-run, it must be re-triggered (CONTinuous mode) or re-initiated and re-triggered.

#### SOURce:SQUare:PERiod

---

Description	Defines the period for the square wave function generator. Overrides a previously set FREQuency.
Syntax	SOURce[n]:SQUare:PERiod[?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: Specifies the period in seconds. Minimum value (s): 2e-6 Maximum value (s): 3600 Default (s): 0.001
Query response	Returns the period (in seconds) for the queried generator or a list of periods in case multiple channels are queried. The reported period will be the same as the value set by the user and <i>not</i> the actual outputted period in case it has been rounded (see notes).
Example(s)	>SOUR2:SQU:PER 0.0001 - sets the square wave period on BNC2 to 0.1ms (10kHz). >SOUR:SQU:PER 0.0001, (@1:24) - sets the square wave period on BNC1-24 to 0.1ms. >SOUR2:SQU:PER? - returns the period of the square wave generator on BNC2. 0.0001 >SOUR:SQU:PER? (@2,3,10) - returns the square wave period of BNCs 2,3 and 10. 0.0001, 0.0001, 0.00002
Notes	As a sample is output every micro-second it is not possible to faithfully reproduce any random square waveform. For example, for a 50% duty-cycle the minimum number of points required per period is 2. The second smallest number of points required for producing a symmetric waveform is 4, and so on adding 2 points each time. For duty-cycles different from 50% it is a bit more subtle. Please see the block about <i>Waveform distortion and magic periods and duty-cycles</i> in section 5.2.2. The generated curve is smoothed by the low-pass filters (section 5.1.2).

---

## SOURce:SQUare:FREQuency

---

Description	Defines the frequency for the square wave function generator. Overrides a previously set PERiod.	
Syntax	SOURce[n]:SQUare:FREQuency[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>: Specifies the frequency in Hz. MINimum value (Hz): 0.00027778 (corresponding to a 1 hour period) MAXimum value (Hz): 5e5 Default (Hz): 1000	
Query response	Returns the frequency (in Hz) for the queried generator or a list of frequencies in case multiple channels are queried. The reported frequency will be the same as the value set by the user and not the actual outputted frequency in case the corresponding PERiod has been rounded (see notes).	
Example(s)	<pre>&gt;SOUR2:SQU:FREQ 25000 - sets the square wave frequency on BNC2 to 25kHz. &gt;SOUR2:SQU:FREQ? - returns the frequency of the square wave generator on BNC2. 25000</pre>	
Notes	See note on magic periods giving a faithful reproduction of the waveform under :PERiod.	

---

## SOURce:SQUare:DCYCLE

---

Description	Defines the duty-cycle for the square wave function generator, i.e. the ratio of the duration of the first part of the a period divided by the period..	
Syntax	SOURce[n]:SQUare:DCYCLE[?] {<numeric_value> MINimum MAXimum} [,channel_list]	
Arguments	<numeric_value>: Specifies the duty cycle in percent. Minimum value (%): 1 Maximum value (s): 99 Default (%): 50.0	
Query response	Returns the duty-cycle (in percent) for the queried generator or a list of duty-cycles in case multiple channels are queried. The reported duty-cycle will be the same as the value set by the user and <i>not</i> the actual outputted duty-cycle in case any of the up or down parts of the waveform is not an integer number of samples (see notes).	
Example(s)	<pre>&gt;SOUR2:SQU:DCYC 25 - sets the square wave duty cycle on BNC2 to 25%. &gt;SOUR:SQU:DCYC 20.0, (@3:24) - sets the square wave duty cycle on BNC3-24 to 20% &gt;SOUR2:SQU:DCYC? - returns the duty cycle of the square wave generator on BNC2. 25 &gt;SOUR:SQU:DCYC? (@2,3,10) - returns the square wave duty cycle of BNCs 2,3 and 10 25, 20, 20</pre>	
Notes	See note on magic periods and duty-cycles giving a faithful reproduction of the waveform under :PERiod.	

---

### SOURce:SQUare:COUNT

---

Description	Defines the number of periods which are generated for one trigger.	
Syntax	SOURce[n]:SQUare:COUNT[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>:	integer, number of repetitions. -1 denotes infinite
	INFinite:	Makes the generator run indefinitely
	Minimum value :	-1 same as INFinite
	Maximum value:	$2^{241} - 1$ (16777215)
	Default:	-1
Query response	Returns the number of periods/cycles produced when the generator is started.	
Example(s)	>SOUR2:SQU:COUN 5	- sets the number of repetitions to 5 for the square wave
	>SOUR2:SQU:COUN?	- returns the number of repetitions
	5	
Notes		

---

### SOURce:SQUare:NCLeft

---

Description	Queries the number of remaining cycles which are generated for one trigger.	
Syntax	SOURce[n]:SQUare:NCLeft? [,channel_list]	
Arguments	none	
Query response	Returns the number of remaining periods/cycles produced of a running generator, including the current cycle. Right after triggering NCLeft will return COUNT and will return 1 during the last cycle. If the generator is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned. After being triggered if COUNT is INF (-1) then -1 is returned.	
Example(s)	>SOUR2:SQU:nc1?	- returns the number of remaining repetitions for the square wave generator on ch 2
	3	
Notes	This command is primarily intended for making it possible for a driver to re-establish the instrument state approximately, if it becomes disconnected to the instrument. But it can also serve as a means for finding out if the generator has finished running.	

---

### SOURce:SQUare:POLarity

---

Description	Defines the if the positive part of a cycle comes before (normal) or after the negative part.	
Syntax	SOURce[n]:SQUare:POLarity[?] {NORMAL INVerted} [,channel_list]	
Argument	NORMAL	default, the positive signal comes before the negative part
	INVerted:	the negative portion of a period before the positive portion
Query response	Returns the polarity of the square wave generator.	

---

---

Example(s)	<pre>&gt;SOUR2:SQU:POL INV &gt;SOUR2:SQU:POL? INV</pre>	<p>- <i>inverts the polarity of the square wave generator</i></p> <p>- returns the polarity</p>
------------	---	---

Notes

---

### SOURce:SQUare:TYPe

---

Description	Defines whether the voltage SPAN is placed symmetrically around the OFFSet, or only on the positive side or the negative side, in which case the generator resembles a pulse generator	
Syntax	SOURce[n]:SQUare:TYPe [?] {SYMMetric POSitive NEGative} [,channel_list]	
Arguments	SYMMetric	OFFSet defines the middle of SPAN (default).
	POSitive:	OFFSet defines the level of the second half of the curve (positive pulses)
	NEGative:	OFFSet defines the level of the first half of the curve (negative pulses)
Query response	Returns type/orientation of the waveform generated by the square wave generator.	
Example(s)	<pre>&gt;SOUR2:SQU:TYP POS &gt;SOUR2:SQU:TYP? POS</pre>	<p>- <i>sets the waveform to positive pulses</i></p> <p>- returns the current waveform type</p>
Notes	With TYPE = SYMMetric as normal voltage symmetric square wave will be generated. If TYPE = POSitive, positive pulses will be generated. Likewise if TYPE = NEGative, negative pulses will be generated – when POLarity is NORMal.	

---



### SOURce:SQUare:VOLTage:SPAN

---

Description	Defines the peak to peak voltage span of the generated square wave, with the centre being equal to the OFFset (default zero).	
Syntax	SOURce[n]:SQUare[:VOLTage]:SPAN [?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>:	peak to peak voltage, or MINimum or MAXimum
	Minimum value :	0V
	Maximum value:	+20V or +4V, depending on the present range
	Default:	0.2 V
Query response	Returns the top to bottom voltage of the square generator.	
Example(s)	>SOUR2:SQU:SPAN 2	- sets peak to peak amplitude to 2V
	>SOUR2:SQU:SPAN? 2	- returns the peak to peak amplitude
Notes	If the SPAN value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting signal is within the actual voltage range. The signal will simply be clipped if it is.	

---

### SOURce:SQUare:VOLTage:OFFSet

---

Description	Defines the relative offset of the square wave generator signal. If OFFSet = 0 the square wave signal will just swing symmetrically around the source's DC value (including the sum of offsets of any other generators).	
Syntax	SOURce[n]:SQUare[:VOLTage]:OFFSet [?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>:	offset in volts
	Minimum value :	-10 or -2V, depending on the range (plus half the SPAN)
	Maximum value:	+10 or +2V, depending on the range (minus half the SPAN)
	Default:	0
Query response	Returns the added offset to the center position of the square wave signal	
Example(s)	>SOUR2:SQU:OFFS 0.7	- sets the offset to +0.7V for the sq. wave generator on ch 2.
	>SOUR2:SQU:OFFS? 0.7	- returns the relative offset
Notes	If the OFFSet value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting AWG signal is within the actual voltage range. The signal will simply be clipped if it is.	

---

## SOURce:SQUare:VOLTage:SLEW

---

Description	Sets or reads the square wave generator slew rate of a channel. When a finite slew rate is set, the rate at which the voltage can change is limited. Together with SPAN and OFFSet , this effectively defines the rise and fall times at the edges of the waveform.
Syntax	SOURce[n]SQUare[:VOLTage]:SLEW[?] <numeric_value> [,chlist]
Arguments	<numeric_value>: Maximum voltage change rate in Volts per second (V/s) Minimum value : 0.01 Maximum value: 2e7 - 20 V/1 $\mu$ s (same as INF) Default: INF
Query response	Returns the current square wave generator slew rate of the specified channel(s).
Example(s)	>SOUR2:SQU:VOLT:SLEW 200 - sets square wave slew rate for channel 2 to 200 V/s. >SOUR2:SQU:VOLT:SLEW? - returns the set square wave slew rate for channel 2 200
Notes	Depending on the amplitude of the waveform, the set slew rate may or may not influence the generated signal. Ultimately the slew rate is limited by the hardware, i.e. by the choice of low pass filter and current measurement range. <i>A small transient (spike) must be expected to appear when changing the slew rate at non zero voltage outputs.</i>

---

### 7.3.5 Triangle generator

Not that changing any parameter, while the generator is running, will end its current trigger cycle and thereby stop the generator and bring its trigger state back to the *Initiated* state if it is in INITiate:CONTinuous mode or otherwise to is *Idle* state. To re-run, it has to be re-triggered (CONTinuous mode) or re-initiated and re-triggered.

#### SOURce:TRlangle:PERiod

---

Description	Defines the period for the triangle function generator. Overrides a previously set FREQuency.
Syntax	SOURce[n]:TRlangle:PERiod[?] <numeric_value> [,channel_list]
Arguments	<p>&lt;numeric_value&gt;: Specifies the period in seconds.</p> <p>Minimum value (s): 4e-6</p> <p>Maximum value (s): 3600</p> <p>Default (s): 0.001</p>
Query response	Returns the period (in seconds) for the queried generator or a list of periods in case multiple channels are queried. The reported period will be the same as the value set by the user and not the actual outputted period in case it has been rounded (see notes).
Example(s)	<pre>&gt;SOUR2:TRI:PER 0.0001 - sets the triangle period on BNC2 to 0.1ms (10kHz). &gt;SOUR:TRI:PER 0.0001, (@1:24) - sets the triangle period on BNC1-24 to 0.1ms. &gt;SOUR2:TRI:PER? - returns the period of the triangle generator on BNC2. 0.0001 &gt;SOUR:TRI:PER? (@2, 3, 10) - returns the triangle period of BNCs 2,3 and 10. 0.0001, 0.0001, 0.00002</pre>
Notes	As samples are output every one micro-second it is not possible to faithfully reproduce any random triangle waveform. For example, for a 50% duty-cycle the minimum number of points required per period is 4. The second smallest number of points required for producing a symmetric waveform is 8, and so on adding 4 points each time. For duty-cycles different from 50% it is a bit more subtle. Please see the block about <i>Waveform distortion and magic</i> periods and duty-cycles in section 5.2.2. Further, the generated curve is smoothed by the low-pass filters (section 5.1.2).

---

## SOURce:TRlangle:FREQuency

Description Defines the frequency for the triangle function generator. Overrides a previously set PERiod.

Syntax SOURce[n]:TRlangle:FREQuency[?] <numeric\_value> [,channel\_list]

Arguments <numeric\_value>: Specifies the frequency in Hz.  
Minimum value (Hz): 0.00027778 (corresponding to a 1 hour period)  
Maximum value (Hz): 250e5  
Default (Hz): 1000

Query response Returns the frequency (in Hz) for the queried generator or a list of frequencies in case multiple channels are queried. The reported frequency will be the same as the value set by the user and not the actual outputted frequency in case the corresponding PERiod has been rounded (see notes).

Example(s) `>SOUR2:TRI:FREQ 25000` - sets the triangle frequency on BNC2 to 25kHz.  
`>SOUR2:TRI:FREQ?` - returns the frequency of the triangle generator on BNC2.  
`25000`

Notes See note on magic periods giving a faithful reproduction of the waveform under :PERiod.

## SOURce:TRlangle:DCYClE

Description Defines the duty-cycle for the triangle function generator.

Syntax SOURce[n]:TRlangle:DCYClE[?] {<numeric\_value>|MINimum|MAXimum} [,channel\_list]

Arguments <numeric\_value>: Specifies the duty cycle in percent.  
Minimum value (%): 1  
Maximum value (s): 99  
Default (%): 50.0

Query response Returns the duty-cycle (in percent) for the queried generator or a list of duty-cycles in case multiple channels are queried. The reported duty-cycle will be the same as the value set by the user and *not* the actual outputted duty-cycle in case one of the up or down parts of the waveform is *not* an integer number of samples (see notes).

Example(s) `>SOUR2:TRI:DCYC 25` - sets the triangle duty cycle on BNC2 to 25%.  
`>SOUR:TRI:DCYC 20.0, (@3:24)` - sets the triangle duty cycle on BNC3-24 to 20%  
`>SOUR2:TRI:DCYC?` - returns the duty cycle of the triangle generator on BNC2.  
`25`  
`>SOUR:TRI:DCYC? (@2,3,10)` - returns the triangle duty cycle of BNCs 2,3 and 10  
`25, 20, 20`

Notes See note on magic periods giving a faithful reproduction of the waveform under :PERiod. Each part (up, down) of the waveform needs to be an integer number of samples for faithful reproduction.

## SOURce:TRlangle:COUnT

Description Defines the number of periods which are generated for one trigger.

Syntax SOURce[n]:TRlangle:COUnT[?] <numeric\_value> [,channel\_list]

Arguments	<numeric_value>:	integer, number of repetitions. -1 denotes infinite
	INFinite:	Makes the generator run indefinitely
	Minimum value :	-1 same as INFinite
	Maximum value:	$2^{241} - 1$ (16777215)
	Default:	-1
Query response	Returns the number of periods/cycles produced when the generator is started.	
Example(s)	>SOUR2:TRI:COUN 5	- sets the number of repetitions to 5 for the triangle
	>SOUR2:TRI:COUN?	- returns the number of repetitions
	5	
Notes		

### SOURce:TRlangle:NCLeft

Description	Queries the number of remaining cycles which are generated for one trigger.	
Syntax	SOURce[n]:TRlangle:NCLeft? [,channel_list]	
Arguments	none	
Query response	Returns the number of remaining periods/cycles produced of a running generator, including the current cycle. Right after triggering NCLeft will return COUNT and will return 1 during the last cycle. If the generator is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned. If COUNT is -1 (infinite) then -1 is returned.	
Example(s)	>SOUR2:TRI:NCL?	- returns the number of remaining repetitions for the triangle generator on ch 2
	3	
Notes	This command is primarily intended for making it possible for a driver to re-establish the instrument state approximately, if it becomes disconnected to the instrument. But it can also serve as a means for finding out if the generator has finished running.	

### SOURce:TRlangle:POLarity

Description	Defines the if the positive part of a cycle comes before (normal) or after the negative part.	
Syntax	SOURce[n]:TRlangle:POLarity[?] {NORMal INVerted} [,channel_list]	
Argument	NORMal	default, the positive signal comes before the negative part
	INVerted:	the negative portion of a period before the positive portion
Query response	Returns the polarity of the triangle generator.	
Example(s)	>SOUR2:TRI:POL INV	- inverts the polarity of the triangle generator
	>SOUR2:TRI:POL?	- returns the polarity
	INV	
Notes		

### SOURce:TRlangle:VOLTage:SPAN

---

Description Defines the peak to peak voltage span of the generated triangle, with the centre being equal to the OFFset (default zero).

Syntax SOURce[n]:TRIangle[:VOLTage]:SPAN [?] <numeric\_value> [,channel\_list]

Arguments <numeric\_value>: peak to peak voltage, or MINimum or MAXimum  
Minimum value : 0V  
Maximum value: +20V or +4V, depending on the present range  
Default: 0.2 V

Query response Returns the top to bottom voltage of the square generator.

Example(s) >SOUR2:TRI:SPAN 2 - sets peak to peak amplitude to 2V  
>SOUR2:TRI:SPAN?  
2 - returns the peak to peak amplitude

Notes If the SPAN value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting signal is within the actual voltage range. The signal will simply be clipped if it is.

---

## SOURce:TRiangle:VOLTage:OFFSet

---

**Description** Defines the relative offset of the triangle generator signal. If OFFSet = 0 the triangle signal will just swing symmetrically around the source's DC value (including the sum of offsets of any other generators).

**Syntax** SOURce[n]:TRiangle[:VOLTage]:OFFSet [?] <numeric\_value> [,channel\_list]

**Arguments**

<numeric_value>:	offset in volts
Minimum value:	-10 or -2V, depending on the range (plus half the SPAN)
Maximum value:	+10 or +2V, depending on the range (minus half the SPAN)
Default:	0

**Query response** Returns the added offset to the center position of the triangle signal

**Example(s)**

>SOUR2:TRI:OFFS 0.7	- sets the offset to +0.7V for the sq. wave generator on ch 2.
>SOUR2:TRI:OFFS?	- returns the relative offset
0.7	

**Notes** The OFFset is intended for compensation and not as an alternative way of setting a DC value, as it is only applied when the generator is running. If the OFFSet value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting AWG signal is within the actual voltage range. The signal will simply be clipped if it is.

---

## SOURce:TRiangle:VOLTage:SLEW

---

**Description** Sets or reads the triangle generator slew rate of a channel. This is mostly relevant for the initial offset applied when the generator starts (and when it ends), to avoid jumps. If the SLEW rate is lower than  $\text{SPAN} / (\text{PERiod} \times \text{MIN}[\text{DCYCLE}, 100\text{-DCYCLE}]/100\%)$  then a phase lag will be introduced.

**Syntax** SOURce[n]:TRiangle[:VOLTage]:SLEW[?] <numeric\_value> [,chlist]

**Arguments**

<numeric_value>:	Maximum voltage change rate in Volts per second (V/s)
Minimum value :	0.01
Maximum value:	2e7 - 20V/1μs (same as INF)
Default:	INF

**Query response** Returns the current triangle generator slew rate of the specified channel(s).

**Example(s)**

>SOUR2:TRI:VOLT:SLEW 200	- sets triangle slew rate for channel 2 to 200 V/s.
>SOUR2:TRI:SLEW?	- returns the set triangle slew rate for channel 2
200	

**Notes** Ultimately the slew rate is limited by the hardware, i.e. by the choice of low pass filter and current measurement range. A small transient (spike) must be expected to appear when changing the slew rate at non zero voltage outputs.

---

### 7.3.6 AWG generator

Not that changing any parameter, while the generator is running, will end its current trigger cycle and thereby stop the generator and bring its trigger state back to the *Initiated* state if it is in INITiate:CONTInuous mode or otherwise to is *Idle* state. To re-run, it has to be re-triggered (CONTInuous mode) or re-initiated and re-triggered. If no TRACe is DEFIned or if it does not exist an execution error will be produced.

#### SOURce:AWG:DEFine

---

Description	Sets or queries which TRACe to be used for the current AWG for the specified channel.
Syntax	SOURce[n]:AWG:DEFine[?] <quoted string> [,channel_list]
Arguments	<quoted string>: Name of the TRACe to be used for this AWG.
Query response	Returns the name of the TRACe assigned to the current AWG. If no TRACe has been assigned (by DEF) an empty quoted string is returned.
Example(s)	<pre>&gt;SOUR1:AWG:DEF "Mywave" - Assigns the TRACe named "Mywave" to the AWG on ch. 1. &gt;SOUR1:AWG:DEF?          - returns name of the TRACe assigned to channel 1. " Mywave"</pre>
Notes	Unless the AWG is already running, DEFine will <i>not</i> produce an error if the named TRACe is yet not present. But, when AWG is started it will check for the presence of the TRACe and will produce an execution error if the TRACe cannot be found.

---

#### SOURce:AWG:COUNt

---

Description	Defines or queries the number of periods which are generated for one trigger.
Syntax	SOURce[n]:AWG:COUNt[?] <numeric_value> [,channel_list]
Arguments	<p>&lt;numeric_value&gt;: integer, number of repetitions. -1 denotes infinite</p> <p>INFinite: Makes the generator run indefinitely</p> <p>Minimum value : -1 same as INFinite</p> <p>Maximum value: <math>2^{24} - 1</math> (16777215)</p> <p>Default: 1</p>
Query response	Returns the number of periods/cycles produced when the generator is started.
Example(s)	<pre>&gt;SOUR2:AWG:COUN 5      - sets the number of repetitions to 5 for the AWG on ch. 2 &gt;SOUR2:AWG:COUN?     - returns the number of repetitions 5</pre>
Notes	

---



### SOURce:AWG:NCLeft

---

Description	Query only. Queries the number of remaining cycles which are generated for one trigger.
Syntax	SOURce[n]:AWG:NCLeft? [,channel_list]
Arguments	none
Query response	Returns the number of remaining periods/cycles produced of a running generator, including the current cycle. Right after triggering NCLeft will return COUNT and will return 1 during the last cycle. If the generator is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned. After being triggered if COUNT is INF (-1) then -1 is returned.
Example(s)	<pre>&gt;SOUR2:AWG:NCL?</pre> - returns the number of remaining repetitions for the AWG on ch. 2  3
Notes	This command is primarily intended for making it possible for a driver to re-establish the instrument state approximately, if it becomes disconnected to the instrument. But it can also serve as a means for finding out if the generator has finished running.

---

### SOURce:AWG:VOLTage:SCALE

---

Description	Defines or queries the scaling factor applied to amplitudes in the DEFINed TRACe, which will convert the TRACe data to voltages.
Syntax	SOURce[n]:AWG[:VOLTage]:SCALE[?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: Scaling factor (implicit unit V), or MINimum or MAXimum Minimum value: -10 Maximum value: 10 Default: 1
Query response	Returns the scaling factor for the AWG(s) on the selected channel(s).
Example(s)	<pre>&gt;SOUR2:AWG:SCAL 8.8</pre> - applies a scaling factor of 8.8V to the TRACe for the AWG on ch. 2.  <pre>&gt;SOUR2:AWG:SCAL?</pre> - returns the AWG scaling factor 8.8
Notes	If the SCALE value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting signal is within the actual voltage range. The signal will simply be clipped if it is.

---

## SOURce:AWG:VOLTage:OFFSet

---

Description	Defines the relative offset of the AWG signal added to the scaled TRACe amplitudes. If OFFSet = 0 AWG will just output the TRACe values multiplied by the SCALe factor.	
Syntax	SOURce[n]:AWG[:VOLTage]:OFFSet [?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>:	offset in volts
	Minimum value :	-10 (volts), depending on the range
	Maximum value:	+10 (volts), depending on the range
	Default:	0
Query response	Returns the offset added to the SCALed TRACe values on the AWG.	
Example(s)	>SOUR2:AWG:OFFS 0.7	- sets the offset to +0.7V for AWG generator on ch 2.
	>SOUR2:AWG:OFFS? 0.7	- returns the offset for the AWG
Notes	<p>The OFFSet is intended for compensation and not as an alternative way of setting a DC value, as it is only applied when the generator is running.</p> <p>If the OFFSet value is set outside its MAXimum and MINimum values, a value error will be reported. However, there is no check if the resulting AWG signal is within the actual voltage range. The signal will simply be clipped if it is.</p>	

---

## SOURce:AWG:VOLTage:SLEW

---

Description	Sets or reads the AWG generator slew rate of a channel.	
Syntax	SOURce[n]AWG[:VOLTage]:SLEW[?] <numeric_value> [,chlist]	
Arguments	<numeric_value>:	Maximum voltage change rate in Volts per second (V/s)
	Minimum value :	0.01
	Maximum value:	2e7 - 20V/1 $\mu$ s (same as INF)
	Default:	INF
Query response	Returns the current AWG generator slew rate of the specified channel(s).	
Example(s)	>SOUR2:AWG:VOLT:SLEW 200	- sets AWG slew rate for channel 2 to 200 V/s.
	>SOUR2:AWG:SLEW? 200	- returns the set AWG slew rate for channel 2
Notes	<p>Depending on the amplitude of the waveform, the set slew rate may or may not have an effect on the generated signal.</p> <p>Ultimately the slew rate is limited by the hardware, i.e. by the choice of low pass filter and current measurement range.</p> <p><i>A small transient (spike) must be expected to appear when changing the slew rate at non zero voltage outputs.</i></p>	

---

## 7.3.7 Trace (AWG) curve management

### TRACe:DEFine

---

Description	Defines and names a new trace and allocates memory for it. Up to 24 traces can be defined.	
Syntax	TRACe:DEFine "quoted_name" , <size>	
Arguments	"quoted_name" :	Name of trace in quotes, maximum 16 ASCII characters.
	<size>:	Size in number of data points for the trace (even positive integer), MINimum 4, MAXimum 6,291,456 )
Query response	None, set only command.	
Example(s)	>TRAC:DEF "Mytrace1",1000    - Defines a trace named Mytrace1 and allocates 1000 datapoints in memory.	
Notes	The device will allocate trace memory corresponding to 4 bytes per data point. If the DEFine will exceed the amount of available memory an error "-225 Out of memory" is produced. If an uneven number of points are defined, <a href="#">in the current version of the firmware (13-1.57)</a> , the last point will be duplicated instead of generating an error.	

---

### TRACe:CATalog

---

Description	Returns a list of names of all traces in memory.	
Syntax	TRACe:CATalog?	
Arguments	None	
Query response	Query returns a comma-separated list of quoted strings containing the names of all traces.	
Example(s)	>TRAC:CAT? "Mytrace1", "Mytrace2"    - Returns catalogue of traces	
Notes	If there are no <trace_name> defined, a single empty string is returned.	

---

## TRACe:REMove:ALL

---

Description	Deletes all traces and frees up the trace memory.	
Syntax	TRACe:REMove:ALL	
Arguments	None	
Query response	None, set only command.	
Example(s)	<code>&gt;TRAC:REM:ALL</code>	<i>- Deletes all traces.</i>
Notes	Deletes all traces and frees up the trace memory. If any trace is currently assigned to a generator, an error will be generated, and nothing happens. So, in order to delete a trace it must be disassociated with the channels' AWGs. Because of memory fragmentation, deleting traces in order to make space for longer traces may not work successfully. Use REM:ALL to reset the entire trace memory.	

---

## TRACe:DATA

---

Description	DATA is used for transferring trace data to the instrument.	
Syntax	TRACe:DATA "quoted_name", <binary block>	
Arguments	"quoted_name" :	Name of defined trace in quotes.
	<binary block>:	IEEE 488.2 binary block (floating point values) (range -1 to 1)
Query response	None, set only command.	
Example(s)	<code>&gt;TRAC:DAT "Mytrace1", #18*****</code>	<i>- sets the TRACe data to the three 4-byte floating point numbers symbolized by the asterisks. Of course an unrealistically short trace.</i>
Notes	When transferring a binary block, the length has to be exactly the same as allocated using the DEFine command. Otherwise an error is produced. When the trace "played" by one of the AWGs, the data will be scaled and offset (into units of Volts) according to the settings on the specific channel's AWG. All data values must be between -1 and 1.	

---

## 7.3.8 Triggering of voltage generators

### SOURce:{ }:ABORt

---

Description	Stops the specified trigger sequence (generator) and brings it to the idle state. This includes setting CONTinuous to OFF to avoid re-triggering.	
Syntax	SOURce[n]:{<name> ALL}:ABORt [,channel_list]	
Arguments	<name>:	DC, SQUare, SINE, TRIangle, AWG
	{source}:	IMMEDIATE, BUS, HOLD, EXTERNAL1,... EXTERNAL5, INTERNAL1,... INTERNAL14
Query response	None. This command is an event.	
Example(s)	>SOUR2:TRI:ABOR	- Stops the triangle generator on channel 2
	>SOUR2:ALL:ABOR	- Stops all generators on channel 2
Notes	<p>For stopping all generators and ongoing current measurements, the general ABORt command can be used.</p> <p>The behavior of setting INIT:CONT to OFF is not according to the standard SCPI protocol, but seems to be more useful.</p> <p>Note that running LIST and SWEep sequences will also be aborted and reset to their starting points. The most recent DC value will, however, stay.</p> <p>Beware that if an output has not reached its set voltage due to slew rate limitations, then the output will continue ramping until the set value has been reached.</p>	

---

### SOURce:{ }:TRIGger:SOURce

---

Description	Sets or reads which trigger source is used for a given generator for the given channel(s).	
Syntax	SOURce[n]:{<name> ALL}:TRIGger:SOURce {source} [,channel_list] SOURce[n]:<name>:TRIGger:SOURce? [,channel_list]	
Arguments	<name>:	DC, SQUare, SINE, TRIangle, AWG
	{source}:	IMMEDIATE, BUS, HOLD, EXTERNAL1,... EXTERNAL5, INTERNAL1,... INTERNAL14
	BUS:	Use the global trigger (*trg command)
	HOLD:	Event detection is disabled for this sequence
	EXTERNAL#:	Use hardware trigger no. #
	INTERNAL#:	Use internal trigger no. #
	IMMEDIATE:	(default) If INITiate:IMMEDIATE is executed, a single trigger event will be processed immediately.
Query response	Returns the current trigger source for the specified channel(s) and generator.	
Example(s)	>SOUR2:TRI:TRIG:SOUR BUS	- sets the trigger source to BUS for the triangle generator on ch. 2
	>SOUR2:SIN:TRIG:SOUR INT1	- sets the trigger source to internal trigger no. 1 for the sine generator on ch. 2
	>SOUR2:SIN:TRIG:SOUR? INT1	- returns the trigger source for channel 2, SINE

---

---

Notes 1) SOUR[n]:ALL:TRIG:SOUR sets or reads the trigger source for all generators  
2) EXTERNAL1..4 are the galvanically isolated BNC trigger input. The other EXT5..10 are reserved for future use on the Digital I/O connector.

---

### SOURce:{ }:INITiate:IMMEDIATE

---

**Description** Initiates the specified generator trigger sequence, making it respond to a single trigger event, where after it returns to its IDLE state (unless CONTinuos is ON).  
This is the simplest way to start a function generator, a sweep or a list sequence for a single run.

**Syntax** SOURce[n]:{<name>|ALL}:INITiate[:IMMEDIATE] [,channel\_list]

**Arguments** <name>: DC, SQUARE, SINE, TRIangle, AWG.

**Query response** None. This command is an event and has no readable state.

**Example(s)**  
 >SOUR2:SIN:TRIG:SOUR IMM - Selects IMMEDIATE as the trigger source for the sine generator on ch. 2.  
 >SOUR2:SIN:INIT - Makes the sine generator on channel 2 wait for a single trigger event. As the trigger source is set to IMMEDIATE, the generator will start when this command is issued.

**Notes** When <name> has SOURce:IMMEDIATE, this command will start <name> immediately. The command has no effect if INIT:CONT is ON.  
When the channel is in FILTER = DC mode, trying to INITiate waveform generators will produce an error, as waveform generators are not available in DC mode.

---

### SOURce:{ }:INITiate:CONTinuous

---

**Description** This command determines whether the trigger sequence for the specified generator is continuously initiated (waiting for trigger) or not. When set to ON the trigger sequence will exit its IDLE state and go to the initiated state waiting for a trigger event. Every time a trigger cycle is completed the generator will go back to the initiated state and wait for a new trigger.

**Syntax** SOURce[n]:{<name>|ALL}:INITiate:CONTinuous {ON|OFF} [,channel\_list]

**Arguments** <name>: DC, SQUARE, SINE, TRIangle, AWG.  
ON: CONTinuous mode is switched on.  
OFF: CONTinuous mode is switched off.

**Query response** Returns the current INITiate:CONTinuous state of the generator.

**Example(s)**  
 >SOUR2:TRI:INIT:CONT ON - Makes the triangle generator on channel 2 start waiting for trigger events. After a completed trigger cycle the generator will wait for a new trigger event.  
 >SOUR2:TRI:INIT:CONT? - Queries the CONTinuous state of the triangle generator on channel 2.  
 ON

---

---

Notes If TRIGger:SOURce is IMMEDIATE the generator will be re-triggering indefinitely until an \*rst or ABORt message is received or CONT is turned OFF.  
When set to OFF an ongoing trigger cycle is allowed to end before the generator goes to its IDLE state.  
When the channel is in FILter = DC mode, trying to INITiate waveform generators will produce an error, as waveform generators are not available in DC mode.

---

## SOURce:{ }:DELay

---

Description Specifies a delay from the trigger event until the generator is actually started.

Syntax SOURce[n]:{<name>|ALL}:DELay[?] <numeric\_value> [,channel\_list]

Arguments <name>: DC, SQUare, SINE, TRIangle, AWG.  
<numeric\_value>: Delay in units of seconds  
Minimum value : 0 (default)  
Maximum value: 3600

Query response Returns the current start delay of the generator.

Example(s) >SOUR2:TRI:DEL 0.02 - Makes the triangle generator on channel 2 start 20 ms after being triggered.  
>SOUR2:TRI:DEL? - Returns the start delay of the triangle generator on ch. 2.  
0.02

Notes

---

## 7.4 Commands for controlling triggers

### 7.4.1 Internal triggers and markers

Internal triggers are either tripped by a generator marker event or by a manual (programmatic) trigger signal.

#### SOURce:{ }:MARKer:STARt:TNUMber

---

Description	Sets which internal trigger should be paired with the STARt MARKer event for the specified generator. The STARt MARKer occurs when the specified generator starts.								
Syntax	SOURce[n]:{<name>}:MARKer:STARt[:TNUMber][?] <numeric_value>								
Arguments	<table border="0"> <tr> <td style="padding-right: 10px;">&lt;name&gt;:</td> <td>DC, SQUare, SIne, TRlangle, AWG.</td> </tr> <tr> <td>&lt;numeric_value&gt;:</td> <td>Internal trigger number (positive integer)</td> </tr> <tr> <td>Minimum value :</td> <td>0* (default)</td> </tr> <tr> <td>Maximum value:</td> <td>14</td> </tr> </table>	<name>:	DC, SQUare, SIne, TRlangle, AWG.	<numeric_value>:	Internal trigger number (positive integer)	Minimum value :	0* (default)	Maximum value:	14
<name>:	DC, SQUare, SIne, TRlangle, AWG.								
<numeric_value>:	Internal trigger number (positive integer)								
Minimum value :	0* (default)								
Maximum value:	14								
Query response	Returns the currently internal trigger number paired with the specified marker.								
Example(s)	<table border="0"> <tr> <td style="padding-right: 10px;">&gt;SOUR2:SQU:MARK:STAR 3</td> <td>- Pairs the STARt marker of the square generator on channel 2 with internal trigger number 3.</td> </tr> <tr> <td style="padding-right: 10px;">&gt;SOUR2:SQU:MARK:STAR?</td> <td>- Returns the internal trigger number which the STARt MARKer on of the square generator on ch. 2 is connected to.</td> </tr> <tr> <td></td> <td>3</td> </tr> </table>	>SOUR2:SQU:MARK:STAR 3	- Pairs the STARt marker of the square generator on channel 2 with internal trigger number 3.	>SOUR2:SQU:MARK:STAR?	- Returns the internal trigger number which the STARt MARKer on of the square generator on ch. 2 is connected to.		3		
>SOUR2:SQU:MARK:STAR 3	- Pairs the STARt marker of the square generator on channel 2 with internal trigger number 3.								
>SOUR2:SQU:MARK:STAR?	- Returns the internal trigger number which the STARt MARKer on of the square generator on ch. 2 is connected to.								
	3								
Notes	<p>*Trigger numbers are 1-14. Zero means <i>no-trigger</i>.</p> <p>There is a one-to-one correspondence between internal triggers and marker events. However, there is (currently) no command for querying which MARKer event a specific internal trigger is connected to, only the reverse.</p>								

---

#### SOURce:{ }:MARKer:END:TNUMber

---

Description	Sets which internal trigger should be paired with the END MARKer event for the specified generator. The END MARKer occurs when the specified generator ENDS (after completing its specified periods, COUNT, or when ABORted).								
Syntax	SOURce[n]:{<name>}:MARKer:END[:TNUMber][?] <numeric_value>								
Arguments	<table border="0"> <tr> <td style="padding-right: 10px;">&lt;name&gt;:</td> <td>DC, SQUare, SIne, TRlangle, AWG.</td> </tr> <tr> <td>&lt;numeric_value&gt;:</td> <td>Internal trigger number (positive integer) or zero</td> </tr> <tr> <td>Minimum value :</td> <td>0* (default)</td> </tr> <tr> <td>Maximum value:</td> <td>14</td> </tr> </table>	<name>:	DC, SQUare, SIne, TRlangle, AWG.	<numeric_value>:	Internal trigger number (positive integer) or zero	Minimum value :	0* (default)	Maximum value:	14
<name>:	DC, SQUare, SIne, TRlangle, AWG.								
<numeric_value>:	Internal trigger number (positive integer) or zero								
Minimum value :	0* (default)								
Maximum value:	14								
Query response	Returns the currently internal trigger number paired with the specified marker.								
Example(s)	<table border="0"> <tr> <td style="padding-right: 10px;">&gt;SOUR2:SQU:MARK:END 4</td> <td>- Pairs the END marker of the square generator on channel 2 with internal trigger number 4.</td> </tr> <tr> <td style="padding-right: 10px;">&gt;SOUR2:SQU:MARK:END?</td> <td>- Returns the internal trigger number which the END MARKer on of the square generator on ch. 2 is connected to.</td> </tr> <tr> <td></td> <td>4</td> </tr> </table>	>SOUR2:SQU:MARK:END 4	- Pairs the END marker of the square generator on channel 2 with internal trigger number 4.	>SOUR2:SQU:MARK:END?	- Returns the internal trigger number which the END MARKer on of the square generator on ch. 2 is connected to.		4		
>SOUR2:SQU:MARK:END 4	- Pairs the END marker of the square generator on channel 2 with internal trigger number 4.								
>SOUR2:SQU:MARK:END?	- Returns the internal trigger number which the END MARKer on of the square generator on ch. 2 is connected to.								
	4								

---



---

Notes \*Trigger numbers are 1-14. Zero means *no-trigger*.  
There is a one-to-one correspondence between internal triggers and marker events.  
However, there is (currently) no command for querying which MARKer event a specific internal trigger is connected to, only the reverse.

---

### SOURce:{ }:MARKer:PStart:TNUMBER

---

**Description** Sets which internal trigger should be paired with the PStart MARKer event for the specified generator.  
The PStart MARKer event occurs whenever a new cycle of one of the waveform generators begins.

**Syntax** SOURce[n]:{<name>}:MARKer:PStart[:TNUMBER][?] <numeric\_value>

**Arguments** <name>: DC, SQUARE, SINE, TRIANGLE, AWG.  
<numeric\_value>: Internal trigger number (positive integer) or zero  
Minimum value : 0\* (default)  
Maximum value: 14

**Query response** Returns the currently internal trigger number paired with the specified marker.

**Example(s)** >SOUR2:DC:MARK:PStart 5 - Pairs the PStart marker of the DC generator on channel 2 with internal trigger number 5.  
>SOUR2:DC:MARK:PST? - Returns the internal trigger number which the PStart MARKer of the DC generator on ch. 2 is connected to.  
5

**Notes** \*Trigger numbers are 1-14. Zero means *no-trigger*.  
There is a one-to-one correspondence between internal triggers and marker events.  
However, there is (currently) no command for querying which MARKer event a specific internal trigger is connected to, only the reverse.

---

### SOURce:{ }:MARKer:PEND:TNUMBER

---

**Description** Sets which internal trigger should be paired with the PEND MARKer event for the specified generator.  
The PEND MARKer event occurs whenever a cycle of one of the waveform generators ends.

**Syntax** SOURce[n]:{<name>}:MARKer:PEND[:TNUMBER][?] <numeric\_value>

**Arguments** <name>: DC, SQUARE, SINE, TRIANGLE, AWG.  
<numeric\_value>: Internal trigger number (positive integer) or zero  
Minimum value : 0\* (default)  
Maximum value: 14

**Query response** Returns the currently internal trigger number paired with the specified marker.

**Example(s)** >SOUR2:DC:MARK:PEND 6 - Pairs the PEND marker of the DC generator on channel 2 with internal trigger number 6.  
>SOUR2:DC:MARK:PEND? - Returns the internal trigger number which the PEND MARKer of the DC generator on ch. 2 is connected to.  
6

---

---

Notes \*Trigger numbers are 1-14. Zero means *no-trigger*.  
There is a one-to-one correspondence between internal triggers and marker events.  
However, there is (currently) no command for querying which MARKer event a specific internal trigger is connected to, only the reverse.

---

### SOURce:{ }:MARKer:SStart:TNUMber

---

Description StepStart marker. Sets which internal trigger should be paired with the SStart MARKer event for the DC generator (Fixed, LIST and SWEEp).  
The SStart MARKer event occurs when a new value of the DC generator is set and is usually used in LIST or SWEEp mode.

Syntax SOURce[n]:DC:MARKer:SStart[:TNUMber][?] <numeric\_value>

Arguments <numeric\_value>: Internal trigger number (positive integer) or zero  
Minimum value : 0\* (default)  
Maximum value: 14

Query response Returns the currently internal trigger number paired with the specified marker.

Example(s) >SOUR2:DC:MARK:SSt 5 - Pairs the DC generator SStart marker on channel 2 with internal trigger number 5.  
>SOUR2:DC:MARK:SSt? - Returns the internal trigger number which the SStart MARKer of the DC generator on ch. 2 is connected to.  
5

Notes \*Trigger numbers are 1-14. Zero means *no-trigger*.  
There is a one-to-one correspondence between internal triggers and marker events.  
However, there is (currently) no command for querying which MARKer event a specific internal trigger is connected to, only the reverse.  
For ANALog SWEEps the StArt marker coincide with the PStArt marker.

---

### SOURce:{ }:MARKer:SEND:TNUMber

---

Description Step END marker. Sets which internal trigger should be paired with the SEND MARKer event for the DC generator (LIST and SWEEp).  
The SEND MARKer event occurs when dwell time is reached for a DC generator iteration in LIST or SWEEp mode.

Syntax SOURce[n]:DC:MARKer:SEND[:TNUMber][?] <numeric\_value>

Arguments <numeric\_value>: Internal trigger number (positive integer) or zero  
Minimum value : 0\* (default)  
Maximum value: 14

Query response Returns the currently internal trigger number paired with the specified marker.

Example(s) >SOUR2:DC:MARK:SEND 6 - Pairs the DC generator SEND marker on channel 2 with internal trigger number 6.  
>SOUR2:DC:MARK:SEND? - Returns the internal trigger number which the SEND MARKer of the DC generator on ch. 2 is connected to.  
6

---

---

Notes \*Trigger numbers are 1-14. Zero means *no-trigger*.  
There is a one-to-one correspondence between internal triggers and marker events.  
However, there is (currently) no command for querying which MARKer event a specific internal trigger is connected to, only the reverse.  
For ANALog SWEeps the SEND marker coincide with the PEND marker.

---

## TINT:SIGNal

---

Description Fires/signals one of the 14 manual (internal) triggers.

Syntax TINT[:SIGNal] <numeric\_value>

Arguments <numeric\_value>: Internal trigger number. Integer in the range 1-14.

Query response N/A

Example(s) >SOUR:SIN:TRIG:SOUR INT2, (@1:24) - sets the trigger source to INTernal1 for sine generators on all channels.  
>TINT 2 - Fires internal trigger #2 starting all square wave generators

Notes

---

## 7.4.2 Trigger output configuration

The trigger output connectors on the front and rear panels are paired up with internal triggers and configured using the following commands.

### OUTPut:TRIGger:SOURce

---

Description	Sets or reads which trigger is routed to the specified trigger output port.	
Syntax	OUTPut:TRIGger[m]:SOURce[?] {BUS, HOLD, EXTernal1, EXTernal2, ..., INTernal1, INTernal2, ...} [,trigger_list]	
Arguments	m:	Output trigger number (1..10)
	{..}	Specifies which trigger the output is connected to: INTernal1, INTernal2, ... (INT1, INT2, ...): Internal triggers EXTernal1, EXTernal2,.. (EXT1, EXT2,...): Trigger In 1..5 BUS: Responding to *trg HOLD: The output is disabled (default)
Query response	Returns the name of the internal trigger or external input trigger currently connected to the specified trigger output.	
Example(s)	<pre>&gt;OUTP:TRIG2:SOUR?      - Queries which trigger is connected to Trigger Out 2. HOLD &gt;OUTP:TRIG2:SOUR INT1  - Connects Trigger Out 2 to internal trigger no. 1 &gt;OUTP:TRIG2:SOUR?      - Queries which trigger is connected to Trigger Out 2. INT1</pre>	
Notes	To manually send out a trigger pulse on one of the <i>Trigger Out</i> ports, first connect it to an internal trigger, and then signal the internal trigger using the TINT[:SIGNAL] command. This is for example needed when aligning the sample output on synchronized instruments.	

---

### OUTPut:TRIGger:WIDTh

---

Description	Sets or reads the width of the trigger pulser appearing on the specified trigger output port.	
Syntax	OUTPut:TRIGger[m]:WIDTh[?] <numeric_value> [,trigger_list]	
Arguments	<numeric_value>:	Trigger pulse width in seconds
	MINimum:	1e-6
	MAXimum:	3600
	Default:	10e-6 (10 $\mu$ s)
	m:	Output trigger number (1..5)
	[,trigger_list]:	List of trigger outputs (for setting multiple triggers)
Query response	Returns the width of the trigger pulse appearing on the specified trigger output(s).	
Example(s)	<pre>&gt;OUTP:TRIG2:WIDT?      - Reads the pulse width on Trigger Out 2. 0.001 &gt;OUTP:TRIG2:WIDT 20e-6 - Set the pulse width on Trigger Out 2 to 20<math>\mu</math>s.</pre>	
Notes	When setting the pulse width, the <numeric_value> will be rounded to the nearest number of integer micro seconds	

---

## OUTPut:TRIGger:POLarity

---

Description	Sets or reads the polarity of the trigger pulser appearing on the specified trigger output port.	
Syntax	OUTPut:TRIGger[m]:POLarity {NORMal INVerted} [,trigger_list]	
Arguments	NORMal:	(default) Trigger pulse goes from low to high (positive going)
	INVerted:	Trigger pulse goes from high to low (negative going)
	m:	Output trigger number (1..5)
	[,trigger_list]:	List of trigger outputs (for setting multiple triggers)
Query response	Returns the polarity of the trigger pulse appearing on the specified trigger output(s).	
Example(s)	<pre>&gt;OUTP:TRIG2:POL? NORM</pre> <i>- Reads the pulse width on Trigger Out 2.</i> <pre>&gt;OUTP:TRIG2:POL INV</pre> <i>- Set the polarity on Trigger Out 2 to negative going.</i>	
Notes	<p>Note when the trigger signal polarity is changed, the trigger output level will immediately change. This may be interpreted as a trigger event for listening devices.</p> <p>Trig Out 1-3 will in NORMal mode have in idle level of 0V and generate pulses of 5V, and opposite in INVerted mode. For Trig Out 4-5 the levels are 0 V and 3.3 V, respectively.</p>	

---

## OUTPut:TRIGger:DELay

---

Description	Sets or reads the delay from the source triggers until the trigger pulser appears on the specified trigger output port(s).	
Syntax	OUTPut:TRIGger[m]:DELay[?] <numeric_value> [,trigger_list]	
Arguments	<numeric_value>:	Trigger delay in seconds
	MINimum:	0
	MAXimum:	3600
	Default:	0
	m:	Output trigger number (1..5)
	[,trigger_list]:	List of trigger outputs (for setting multiple triggers)
Query response	Returns the delay of the trigger pulse appearing on the specified trigger output(s).	
Example(s)	<pre>&gt;OUTP:TRIG2:DEL? 0.1e-3</pre> <i>- Reads the pulse delay (of 0.1ms) on Trigger Out 2.</i> <pre>&gt;OUTP:TRIG2:DEL 50e-6</pre> <i>- Set the pulse delay on Trigger Out 2 to 50µs.</i>	
Notes	When setting the pulse delay, the <numeric_value> will be rounded to the nearest number of integer micro seconds	

---

## OUTPut:SYNChronize:SIGNal

---

Description	Makes the instrument go into SYNC Wait mode and outputs a SYNC pulse on the specified Trigger Out 4 / Sync port. Used for synchronizing multiple QDAC-II units
Syntax	OUTPut:SYNChronize:SIGNal
Arguments	None
Query response	None. This is an event.

---

---

Example(s) `>OUTP:SYNC:SIGN` *Sends out a sync pulse on Trigger Out 4 / Sync.*

Notes This event sends a SYNC signal out on Trigger Out #4 (Sync). The signal will have positive polarity.

---

## 7.5 Current sensor commands

### SENSe:CURRent:ABORt

---

Description	Stops the specified current sensor trigger sequences and brings them to their idle state, thus also setting INIT:CONTinuous to OFF*.
Syntax	SENSe[n][:CURRent]:ABORt [,channel_list]
Arguments	None
Query response	None. This command is an event.
Example(s)	>SENS2:ABOR - Stops current sensor measurements on channel 2
Notes	This command will not just abort the triggering sequence, it will also stop an ongoing already triggered measurement, which can involve several COUNTs and channels. For stopping all generators and ongoing current measurements, the general ABORt command can be used. The effect for the current sensors is the same. *)The behavior of setting INIT:CONT to OFF is not according to the standard SCPI protocol but is more user friendly.

---

### SENSe:CURRent:TRIGger:SOURce

---

Description	Sets or reads which trigger source is used by the current sensor for the given channel(s).
Syntax	SENSe[n][:CURRent]:TRIGger:SOURce {source} [,channel_list] SENSe[n][:CURRent]:TRIGger:SOURce? [channel_list]
Arguments	{source}: IMMEDIATE, BUS, HOLD, EXTERNAL1,... EXTERNAL5, INTERNAL1,... INTERNAL14 BUS: Use the global trigger (*trg command) HOLD: Event detection is disabled for this sequence EXTERNAL#: Use hardware trigger no. # INTERNAL#: Use internal trigger no. # IMMEDIATE: (default) If INITiate:IMMEDIATE is executed, a single trigger event will be processed immediately. Automatic triggering will not occur by itself.
Query response	Returns the current trigger source for current sensor(s) of the specified channel(s).
Example(s)	>SENS2:TRIG:SOUR BUS - sets the current sensor trigger source on ch. 2 to BUS. >SENS2:TRIG:SOUR INT1 - sets the current sensor trigger source on ch. 2 to internal trigger no. 1. >SENS2:TRIG:SOUR? INT1 - returns the current sensor trigger source for ch. 2.
Notes	The default trigger source for the current sensor is IMMEDIATE, meaning that SENS[n]:INIT will trigger a measurements.

---

## SENSe:CURRent:INITiate:IMMEDIATE

---

Description	Initiates the current sensor trigger sequence for the specified channel(s), making it respond to a single trigger event, where after it returns to its IDLE state (unless CONTInuos is ON). This command clears the measurement buffer.	
Syntax	SENSe[n]:[CURRent]:INITiate[:IMMEDIATE] [,channel_list]	
Arguments	None	
Query response	None. This command is an event and has no readable state.	
Example(s)	<pre>&gt;SENS2:TRIG:SOUR IMM &gt;SENS2:INIT</pre>	<p>- Selects IMMEDIATE as the trigger source for current sensor on ch. 2.</p> <p>- Makes the current sensor on channel 2 wait for a single trigger event. As the trigger source is set to IMMEDIATE, the current sensor value will be captured right when this command is issued.</p>
Notes	<p>If the trigger source is IMMEDIATE a single measurement is performed (COUNT points).</p> <p>Note that in order to record and buffer multiple measurements one should either trigger multiple times in CONT=ON mode or set COUNT &gt; 1 as the INIT:IMM command clears the measurement buffer as the first thing.</p> <p>The command has no practical effect if INIT:CONT is ON.</p>	

---

## SENSe:CURRent:INITiate:CONTInuous

---

Description	<p>This command determines whether the trigger sequence for current sensors are <i>continuously</i> initiated (waiting for a trigger) or not. When set to ON the trigger sequence will exit its IDLE state and go to the initiated state waiting for a trigger event. Every time a trigger cycle is completed the sensor trigger sequence will go back to the initiated state and wait for a new trigger.</p> <p>When INIT:CONT is switched from OFF to ON the measurement buffer is cleared.</p>	
Syntax	SENSe[n]:[CURRent]:INITiate:CONTInuous {ON OFF} [,channel_list]	
Arguments	<p>ON: CONTInuous mode is switched on.</p> <p>OFF: CONTInuous mode is switched off.</p>	
Query response	Returns the current INITiate:CONTInuous state of the current sensor(s).	
Example(s)	<pre>&gt;SENS2:INIT:CONT ON &gt;SENS2:INIT:CONT? ON</pre>	<p>- Makes the current sensor on channel 2 start waiting for trigger events. After a completed trigger cycle the sensor will wait for a new trigger event.</p> <p>- Queries the CONTInuous state of the current sensor on channel 2.</p>
Notes	<p>If TRIGger:SOURce is IMMEDIATE the sensor will be re-triggering indefinitely until an *rst or ABORt message is received or CONT is turned OFF.</p> <p>When set to OFF an ongoing trigger cycle, i.e. an already triggered measurement, is allowed to end before the sensor trigger sequence goes to its IDLE state.</p> <p>Note that if TRIGger:SOURce is IMMEDIATE and CONT = ON then there will be only one measurement per A/D converter sample interval (0.33.. ms). If APERture is greater than the sample interval, then the integration periods will be overlapping.</p>	

---



## SENSe:CURRent:DELay

---

Description	Specifies a delay from the trigger event until the integrated current sensor value is captured. The applied value will be rounded to nearest multiple of sample intervals.
Syntax	SENSe[n][:CURRent]:DELay[?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: Delay in seconds in increments of 0.33.. ms Minimum value : 0 (default) Maximum value: 3600
Query response	Returns the measurement delay of the current sensor. As the current sensor integrated value is updated only every 0.33.. ms, there is an uncertainty of up to half a current sensor sample interval on the returned value.
Example(s)	<pre>&gt;SENS2:DEL 0.0278</pre> - Sets a delay of 27.8 ms. However, this value will be rounded to 28ms (84 x 0.33.. ms) <pre>&gt;SENS2:DEL?</pre> - Returns the start delay of the current sensor on ch. 2. <pre>0.028</pre>
Notes	The delay mechanism can for example be used to let allow for the current sensor to integrate over an entire APERTure after fx a triggered change in voltage, before the value is captured. Note that the current sensor has a digital filter with a time constant of about 1.5ms. to be on the save side minimum delay of 3 ms should be used. As the current sensor is probed every 0.33 ms the actual delay will always me an integer number of the update period.

---

## SENSe:CURRent:COUNt

---

Description	Specifies how many sensor readings (each of APERTure length in time) should be performed upon a single trigger event.
Syntax	SENSe[n][:CURRent]:COUNt[?] <numeric_value> [,channel_list]
Arguments	<numeric_value>: Number of readings (integer). Minimum value : 1 Maximum value: 65635
Query response	Returns the number of readings to be performed upon a single trigger event.
Example(s)	<pre>&gt;SENS2:COUN 5</pre> - Causes the current sensor on channel 2 to make 5 readings for each trigger event.. <pre>&gt;SENS2:COUN?</pre> - Returns the number of current readings generated per trigger event on ch. 2. <pre>5</pre>
Notes	If COUNt = 1 (default), then the present value integrated for the last APERTure seconds is captured immediately. If COUNt > 1, then the first reading will be immediate, but the following readings will be spaced by APERTure seconds in time. If multiple triggers have been received and COUNt > 1 then readings will be arranged in chronological order. Say that COUNt = 2 and that 3 triggers have been received then the stored data will be: reading1, reading1, reading2, reading2, reading3, reading3

---

## SENSe:CURRent:NCLeft

---

Description	Queries how many sensor readings are remaining since a trigger event.	
Syntax	SENSe[n]:[CURRent]:NCLeft? [,channel_list]	
Arguments	None, query only.	
Query response	Returns the remaining number of readings to be performed upon a single trigger event.	
Example(s)	>SENS2:NCLeft?	- Returns the number of remaining current readings for the most recent trigger events on ch. 2.
	4	
Notes	Returns the number of remaining readings for the specified current sensor(s) including the present reading. Right after triggering NCLeft will return COUNT, but as the current sensor always responds with the first reading immediately, it will, unless a trigger delay is specified, return COUNT-1. While averaging the last reading in a series of COUNT, NCLeft will report 1. If the sensor is in its IDLE mode or is INITiated but not triggered a NCLeft value of zero is returned.	

---

## SENSe:CURRent:RANGe

---

Description	Sets or reads the current range (measurement and output range) for individual channels.	
Syntax	SENSe[n]:[CURRent]:RANGe[?] {LOW HIGH} [,channel_list]	
Arguments	LOW:	Low current range ( $\pm 200\text{nA}$ ).
	HIGH:	High current range ( $\pm 10\text{mA}$ ) (default).
Query response	Returns the current output voltage range for the selected channel(s).	
Example(s)	>SENS2:RANG LOW	- sets the current range for channel 2 to LOW.
	>SENS2:RANG?	- returns the present current range of channel 2
	LOW	
Notes	When switching range, a relay in the output path is toggled and switches in another resistor. A feedback loop will adjust the voltage in case this results in a change of current. A transient may occur. So, it is advised to do this predominantly at zero output voltage. The actual electromechanical switching of range takes about 30-40 ms. Please see the detailed information in section 5.3.1.	

---

## SENSe:CURRent:APERture

---

Description	Sets or reads the current sensor integration time for the specified channel(s). Clears the measurement buffer.	
Syntax	SENSe[n]:CURRent]:APERture[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>: Positive floating point number, specifies the integration time in seconds. Minimum increment is 0.00033... MINimum value: 0.00033.. (0.33.. ms) MAXimum value: 2 (2 seconds) DEFault: LFR 50Hz: 0.02 (20 ms), LFR 60 Hz: 0.0166.. (16.66.. ms).	
Query response	Returns the integration time for the selected channel(s) in units of seconds. The value will be a multiple of the sampling interval 0.00033.. .	
Example(s)	<pre>&gt;SENS2:APER 0.2</pre> <p>- sets the current sensor integration time to 200ms for ch. 2 (corresponding to 10x 50Hz PLC).</p> <pre>&gt;SENS2:APER?</pre> <p>- returns the present current sensor integration time for channel 2</p> <p>0.2</p>	
Notes	Integration time is the period over which samples from the current sensor analog-to-digital (A/D) converter are integrated for one measurement. The current sensor A/D converter is sampled every 1/3 ms. The result is integrated continuously according to APERture or NPLCycles. This means that when a current measurement is triggered it is the value integrated for the past APERture seconds (integers of A/D samples) which is reported. When changing APERture one should thus wait for one integration time before capturing the measurement. Please see the detailed information in section 5.3.1.	

---

## SENSe:CURRent:NPLCycles

---

Description	Sets or reads the current sensor integration time in numbers of power line cycles for the specified channel(s). Clears the measurement buffer.	
Syntax	SENSe[n]:CURRent]:NPLCycles[?] <numeric_value> [,channel_list]	
Arguments	<numeric_value>: Specifies the number of power line cycles. MINimum value: 0.0166.. (corresponding to a single sample at 50 Hz) MAXimum value: 100 Default: 1	
Query response	Returns the current sensor integration time in number of power cycles for the selected channel(s). This number will be rounded to nearest multiple of 0.05.	
Example(s)	<pre>&gt;SENS2:NPLC 1</pre> <p>- sets the current sensor integration time to 20ms for ch. 2 (when LFRrequency equals 50Hz).</p> <pre>&gt;SENS2:NPLC?</pre> <p>- returns the present number of power line cycles used in current sensor integration for channel 2</p> <p>1</p>	

---

---

**Notes** The current sensor A/D converter is sampled every 0.33.. ms. Therefore, the minimum incremental value is 0.0166.. at 50 Hz. If NPLCycles is set to other than a multiple of 0.0166.. at 50 Hz, it will be rounded to the nearest multiple. No error will be produced. At 60 Hz the minimum increment is 0.02.  
When setting NPLCycles, APERTure will be set to NPLCycles / LFRrequency.  
Please see the comments for [APERTure](#) detailed information in section 5.3.1.

---

## SENSe:CURRent:DATA:REMove

---

<b>Description</b>	Returns and removes previously recorded data from the current-measurement buffers of the selected channels. The data points are results of previously triggered measurements.						
<b>Syntax</b>	SENSe[n]:CURRent]:DATA:REMove? [<num_points>] [,channel_list]						
<b>Arguments</b>	<table border="0"> <tr> <td style="padding-right: 20px;">&lt;num_points&gt;:</td> <td>Number of points to read and remove (integer)</td> </tr> <tr> <td>MINimum:</td> <td>1</td> </tr> <tr> <td>MAXimum:</td> <td>65636</td> </tr> </table>	<num_points>:	Number of points to read and remove (integer)	MINimum:	1	MAXimum:	65636
<num_points>:	Number of points to read and remove (integer)						
MINimum:	1						
MAXimum:	65636						
<b>Query response</b>	<p>Returns the &lt;num_points&gt; oldest readings from the measurement buffer of each channel Returns all measurement data in the measurement buffer in ASCII format and deletes the points from the measurement buffer. If multiple channels are specified by channel_list, then all data from the first channel in the list is returned followed by all data from the second channel in the list etc. separated by commas.</p> <p>If &lt;num_points&gt; is greater than the number of points in the queried measurement buffers, then DATA:REMove will produce an error and will not return any data to the controlling computer.</p> <p>If &lt;num_points&gt; is not provided, then DATA:REMove will return the entire content(s) of the measurement buffer(s) and clear them.</p>						
<b>Example(s)</b>	<pre>&gt;SENS4:DATA:REM? 3 - Returns the three oldest current readings from the current sensor measurement buffer on ch. 4. -3.49783e-10, 1.1638535e-12, -7.4938563e-9</pre>						
<b>Notes</b>	<p>Note that if multiple channels are queried, they may have different number of readings in their buffers. In that case the return result can only be deciphered by knowing the DATA:POINTs of the individual channels.</p> <p>If there are not enough data in the measurement buffer(s) then an error should be raised (for example '-230,"Data corrupt or stale"') and the query will time out on the controlling computer as no data will be returned. Also, no data are cleared from measurement buffers in this case.</p>						

---

## SENSe:CURRent:DATA:LAST

---

Description	Returns the last (newest) measurement from the measurement buffer of the selected channels. No values are cleared from the measurement buffer(s) by this command.
Syntax	SENSe[n]:CURRent]:DATA:LAST? [channel_list]
Arguments	none
Query response	Returns the most recent current measurement from the addressed channel(s) without deleting the measurement point(s). Even if measurement buffers have been cleared for example by DATA:REMove, then DATA:LAST will still report the most recent measurement. If no measurement has been performed since *RST or power up then NaN (not a value) will be returned (numerical 9.91e+37).
Example(s)	>SENS4:DATA:LAST? - Returns the most recent current reading from the current sensor on ch. 4. -3.49783e-10
Notes	

---

## SENSe:CURRent:DATA:POINts

---

Description	Returns the number of data points in the measurement buffer(s) of the selected channel(s).
Syntax	SENSe[n]:CURRent]:DATA:POINts? [channel_list]
Arguments	none
Query response	Returns how many current measurement data points are available in the measurement buffer(s) of the addressed channel(s). If multiple channels are addressed, then a comma separated list of values will be returned (always in ASCII format).
Example(s)	>SENS4:DATA:POIN? - Returns the number of data points in the current measurement buffer of ch. 4. 3
Notes	

---

## READ:CURRent

---

Description	<p>Performs a current measurement for the selected channels, returns the result and clears the measurement buffer. It will also stop any ongoing measurements and reset the trigger system.</p> <p>This command sets the trigger source to IMMEDIATE, and therefore will always answer promptly (unless APERTURE is large and COUNT &gt; 1).</p>
Syntax	READ[n]:CURRent? [channel_list]
Arguments	None
Query response	Returns all measurement data in the measurement buffer in ASCII format. If multiple channels are specified by channel_list, then all data from the first channel in the list is returned followed by all data from the second channel in the channel_list etc. separated by commas.
Example(s)	<p>&gt;READ21? - Returns the present current sensor readout on channel 21. -3.49783e-10</p> <p>&gt;READ? (@1:4) - Returns the present current sensor readout on channels 1-4. -2.978352e-10,1.1638535e-12,-7.4938563e-9, 2.97835e-10</p>
Notes	<p>This command reads present integrated current sensor output(s) for the selected channels and returns the data to the controller. This command does not wait for external or internal triggers.</p> <p>The READ command behind the scenes sends an ABORT message, sets the TRIGGER:SOURCE to IMMEDIATE and issues a followed by an INITIATE:IMMEDIATE message to the current sensor mechanism, and sends a DATA:REMOVE? query message. <i>This implies that if INIT:CONT was on before this command was executed, it will be OFF afterwards and that the trigger source has changed if it was not already IMMEDIATE.</i></p> <p>This command does not restart a current measurement, it just reports the currently integrated value. So, if the integration time is larger than the time since the most recent voltage change, the reported current will not have reached a stable value.</p> <p>Only if COUNT is &gt;1 then after capturing the first value(s) then the command will wait for a time defined by APERTURE before capturing the next values.</p> <p>Note that if COUNT is not the same for all channels in a multi-channel query, the return result can only be deciphered by knowing the COUNT of the individual channels.</p> <p>Unlike for example Keysight DMMs (3446x) the QDAC-II will always enforce a temporary IMMEDIATE trigger source (TRIG:SOUR IMM) instead of reporting an error if INIT:CONT = ON. This is not standard behavior, but believed to be more user friendly.</p> <p><i>READ is a sequential command which may block the communication or time out, if for example COUNT x APERTURE represents a long time. To avoid blocking current measurements use INIT followed by a trigger command (*trg) instead and read the results using FETCH? or DATA:REMOVE?</i></p>

---

## FETCH:CURRent

---

Description	<p>Returns all available data in the measurement buffers of the selected channels. The data points are results of previously triggered measurements. The command will wait for an ongoing measurement, which can be relevant if COUNT &gt; 1.</p> <p>The measurement buffers are not cleared but can be re-read.</p>
Syntax	FETCH[n]:CURRent? [channel_list]

---

---

Arguments	None
Query response	Returns all measurement data in the measurement buffer in ASCII format. If multiple channels are specified by channel_list, then all data from the first channel in the list is returned followed by all data from the second channel in the list etc. separated by commas.
Example(s)	<p>&gt;FETC11?                   - Returns contents of the current sensor measurement buffer on channel 11. In this case three values. -3.49783e-10,1.1638535e-12,-7.4938563e-9</p> <p>&gt;FETC? (@1:4)               - Returns contents of the current sensor measurement buffers on channels 1-4. -2.978352e-10,1.1638535e-12,-7.4938563e-9,-2.97835e-10</p>
Notes	<p>The FETCh command does not clear the measurement buffer. It also does not trigger measurements.</p> <p>If there are no data in the measurement buffer(s) then an error will be produced, and the read operation will time out.</p> <p>Note that if COUNT is not the same for all channels in a multi-channel query, the return result can only be deciphered by knowing the COUNT of the individual channels.</p>

---

## READ:ADC

---

Description	<p>Same as READ:CURRent except it is the raw integer from the current sensor analog to digital converter which is reported</p> <p>Performs a current measurement for the selected channels, returns the result and clears the measurement buffer.</p> <p>This command momentarily sets the trigger source to IMMEDIATE, and therefore will always answer promptly (unless APERTure is large).</p>
Syntax	READ[n]:ADC? [channel_list]
Arguments	None
Query response	Returns all measurement data in the measurement buffer in ascii format. If multiple channels are specified by channel_list, then all data from the first channel in the list is returned followed by all data from the second channel in the channel_list etc. separated by commas.
Example(s)	<p>&gt;READ21:ADC?               - Returns the present raw current sensor readout on channel 21. 262199</p>
Notes	<p>This command triggers the transfer of the present integrated current sensor output(s) for the selected channels and returns the data to the controller. This command does not wait for external or internal triggers. <u>Unlike READ:CURRent? this command only returns a single value even if COUNT &gt; 1.</u></p> <p>The reported values are the raw integers from the A/D converters before any applied scaling and offsetting (using calibration parameters, see DIAG:ICAL. The command is primarily used for calibration purposes.</p> <p>See also notes for READ:CURRent.</p>

---

## 7.6 System commands

System commands are related to general instrument configuration and reading the error message queue and not controlling the primary functions of the instrument (generating voltages and measuring currents).

### 7.6.1 Communication setup commands

#### SYSTem:COMMunicate:LAN:DHCP

---

Description	Enables, disables, or queries the instrument's use of DHCP (Dynamic Host Configuration Protocol) for getting its IP address, subnet mask and default gateway.	
Syntax	SYSTem:COMMunicate:LAN:DHCP[?] {OFF 0 ON 1}	
Arguments	ON 1:	The instrument tries to obtain an IP address from a DHCP server at power on (default).
	OFF  0:	The instrument uses the static IP address, Subnet Mask, and Default Gateway. The same happens in the absence of a DHCP server.
Query response	Returns the current setting, 0 (OFF) or 1 (ON).	
Example(s)	<pre>&gt;SYST:COMM:LAN:DHCP ON - Enables DHCP. &gt;SYST:COMM:LAN:UPD - Stores any changed settings. &gt;SYST:REST - Restarts the firmware including the LAN interface. &gt;SYST:COMM:LAN:DHCP? - Queries the DHCP state. 1</pre>	
Notes	<p>To store this setting a :LAN:UPDate must be executed. For changes to take effect a SYSTem:REStart (or power cycling) is required.</p> <p>Configuration of the LAN interface is often done via the USB interface, as LAN may not be functional.</p> <p>Note that if the instrument does not have a permanent entry in the DHCP server's map of clients (static DHCP), the retrieved IP address may change between power on events or other events leading to new requests.</p> <p>If no DHCP server is available or DHCP is OFF, the instrument will use the stored IP address, Subnet Mask, and Default Gateway.</p>	

---

#### SYSTem:COMMunicate:LAN:HOSTname

---

Description	Sets or queries the host name of the instrument. This is the name under which the instrument appears on the local network. The factory set host name is identical to the serial number of the unit.	
Syntax	SYSTem:COMMunicate:LAN:HOSTname "<hostname>" SYSTem:COMMunicate:LAN:HOSTname?	
Arguments	"<hostname>"	Quoted string containing the new name, max 16 characters.
Query response	Returns the host name as a quoted string.	
Example(s)	<pre>&gt;SYST:COMM:LAN:HOST? - Queries the unit's LAN host name "Q310-12345"</pre>	

---



---

**Notes** The hostname is the host portion (left most) of the domain name, which is translated into an IP address. When DHCP is activated, the hostname is registered with the Dynamic Domain Name System (Dynamic DNS) service at power-on.

To store this setting a :LAN:UPDate must be executed. For changes to take effect a SYSTem:REStart (or power cycling) is required.

---

### SYSTem:COMMunicate:LAN:MAC?

---

<b>Description</b>	Queries the instrument's Media Access Control (MAC) address	
<b>Syntax</b>	SYSTem:COMMunicate:LAN:MAC?	
<b>Arguments</b>	None	
<b>Query response</b>	Returns the 12-character hexadecimal MAC address surrounded by quotes.	
<b>Example(s)</b>	>SYST:COMM:LAN:MAC? "70B3D5921000"	- Queries the instrument's MAC address.
<b>Notes</b>		

---

### SYSTem:COMMunicate:LAN:IPADdress

---

<b>Description</b>	Sets or queries the instrument's IP address.	
<b>Syntax</b>	SYSTem:COMMunicate:LAN:IPADdress "<address>" SYSTem:COMMunicate:LAN:IPADdress? [{CURRent STATIC}]	
<b>Arguments</b>	"<address>": [{CURRent STATIC}]:	IP address surrounded by quotes. Optional argument. If present, it determines if it is the stored address or the actual address in use which is queried. CURRent is default.
<b>Query response</b>	Returns either the stored (static) IP address or the actually used IP address, which may be different from the stored one if DHCP is ON.	
<b>Example(s)</b>	>SYST:COMM:LAN:IPAD "192.168.14.178" >SYST:COMM:LAN:IPAD? STATIC "192.168.14.178"	-Sets the static IP address -Queries the static IP address.
<b>Notes</b>	Assigns a static Internet Protocol (IP) address for the instrument. If DHCP is enabled (SYSTem:COMMunicate:LAN:DHCP ON), the specified static IP address is not used.  As with other LAN setting commands to store this setting a :LAN:UPDate must be executed. For changes to take effect a SYSTem:REStart (or power cycling) is required.  Further it is not possible to use the short form of the keyword "STATIC", which would be "STAT" .	

---

## SYSTem:COMMunicate:LAN:GATeway

---

Description	Sets or queries the default gateway for the instrument.	
Syntax	SYSTem:COMMunicate:LAN:GATeway "<address>" SYSTem:COMMunicate:LAN:GATeway? [{CURRENT STATIC}]	
Arguments	"<address>":	IP address surrounded by quotes.
	[{CURRENT STATIC}]:	Optional argument. If present, it determines if it is the stored gateway or the current gateway in use which is queried. CURRENT is default.
Query response	Returns either the stored gateway (STATIC) or the actually used gateway (CURRENT), which may be different from the stored one if DHCP is ON.	
Example(s)	>SYST:COMM:LAN:GAT "192.168.1.1"	-Sets the static default gateway.
	>SYST:COMM:LAN:GAT? STAT	-Queries the static default gateway.
	"192.168.1.1"	
Notes	<p>Assigns a static default gateway for the instrument. If DHCP is enabled (SYSTem:COMMunicate:LAN:DHCP ON), the specified static gateway is not used.</p> <p>As with other LAN setting commands to store and activate changes to this setting a SYSTem:COMMunicate:LAN:UPDate must be executed.</p> <p>Further a SYSTem:REStart is required in order for changes to take effect.</p>	

---

## SYSTem:COMMunicate:LAN:SMASK

---

Description	Sets or queries the instrument's sub-net mask.	
Syntax	SYSTem:COMMunicate:LAN:SMASK "<mask>" SYSTem:COMMunicate:LAN:SMASK? [{CURRENT STATIC}]	
Arguments	"<mask>":	Sub-net address surrounded by quotes.
	{CURRENT STATIC}:	Optional argument. If present, it determines if it is the stored mask or the current mask in use which is queried. CURRENT is default.
Query response	Returns either the stored (static) subnet mask or the actually used mask, which may be different from the stored one if DHCP is ON.	
Example(s)	>SYST:COMM:LAN:SMASK "255.255.255.0"	-Sets the static sub-net mask.
	>SYST:COMM:LAN:SMASK? STAT	-Queries the static sub-net mask..
	"255.255.255.0"	
Notes	<p>Assigns a static sub-net mask for the instrument. If DHCP is enabled (SYSTem:COMMunicate:LAN:DHCP ON), the specified static sub-net mask is not used.</p> <p>As with other LAN commands a :LAN:UPDate is required to store the setting in non-volatile memory and a SYSTem:REStart or power cycling is required in order for changes to take effect.</p>	

Implementation notes

---

## SYSTem:COMMunicate:LAN:UPDate

---

Description	Stores the current LAN settings including any changes in non-volatile memory.
Syntax	SYSTem:COMMunicate:LAN:UPDate
Arguments	None
Query response	None. This is an event.
Example(s)	>SYST:COMM:LAN:UPD      - Store LAN settings to non-volatile..
Notes	A SYSTem:REStart is required in order for changes to take effect

---

## 7.6.2 Error system

### SYSTem:ERRor:ALL

---

Description	Reads all error and event messages from the error queue
Syntax	SYSTem:ERRor:ALL?
Arguments	None
Query response	Returns a comma separated list of error codes and descriptions and empties the queue. If the queue is already empty '0, "No error"' is returned. The return format is (oldest first): <error code>,<error string>,<error code>,<error string> ....
Example(s)	>SYST:ERR:ALL?                    - Reads all messages in the error queue and clears the queue. -114,"Header suffix out of range;SOUR36",-113,"Undefined header;SOYR" >SYST:ERR:ALL? 0, "No error"
Notes	

---

### SYSTem:ERRor:NEXT

---

Description	Reads the oldest error message from the error/event queue
Syntax	SYSTem:ERRor[:NEXT]?
Arguments	None
Query response	Returns a comma separated pair of error codes and descriptions and empties the queue. If the queue is empty - 0, "No error" - is returned. The return format is: <error code>,<error string>
Example(s)	>SYST:ERR?                        - Reads and removes the oldest message from the error queue -114,"Header suffix out of range;SOUR36"
Notes	If the error queue is filled and more errors occur up the most recent error in the queue will be replaced by - 350,"Queue overflow".

---

### SYSTem:ERRor:COUNT

---

Description	Reports the number of error messages in the error/event queue
Syntax	SYSTem:ERRor:COUNT
Arguments	None
Query response	Returns an positive integer equal to the number of entries in the error/event queue.<error
Example(s)	>SYST:ERR:COUN?                - Queries the number of messages in the error/event queue 2

---

### 7.6.3 Other commands

#### SYSTem:BEEPer:STAT

---

Description	Sets or reads the current state of the built-in error buzzer.	
Syntax	SYSTem:BEEPer:STAT[?]{ON OFF}	
Arguments	{ON OFF}:	ON: The buzzer sounds when an error occur (default). OFF: The buzzer is silent when errors occur.
Query response	Reports if the buzzer is enabled or disabled for error reporting.	
Example(s)	<pre>&gt;SYST:BEEP:STAT OFF</pre> <i>- Causes the buzzer not to sound when errors occur.</i> <pre>&gt;SYST:BEEP:STAT?</pre> <i>- Queries the buzzer state.</i> <pre>OFF</pre>	
Notes	Sometimes one gets tired of hearing a buzz very time an error occurs. In such cases the buzzer can disabled.	

---

#### SYSTem:BEEPer:IMMEDIATE

---

Description	Makes the built-in buzzer sound.	
Syntax	SYSTem:BEEPer[:IMMEDIATE]	
Arguments	None	
Query response	No query version. Event only.	
Example(s)	<pre>&gt;SYST:BEEP</pre> <i>- Makes the instrument produce a sound.</i>	
Notes		

---

#### SYSTem:TEMPERature

---

Description	Reads the on board temperature sensors	
Syntax	SYSTem:TEMPERature? <board_no>, <sensor_no>	
Arguments	<board_no>:	1, 2, or 3. (1: upper BNC row, 2: middle, 3: lower).
	<sensor_no>:	1, 2, or 3. (1: left, 2: middle, 3: right).
Query response	Query only command. Returns the temperature (in degree Celcius) at the specified location on the specified output printed circuit board (BNC rows 1-8, 9-16, 17-24)	
Example(s)	<pre>&gt;SYST:TEMP? 1,3</pre> <i>- Reads the temperature of the right most sensor on the upper board.</i> <pre>38.91</pre>	
Notes	Used very rarely, primarily for diagnostic purposes.	

---

## SYSTem:CLOCK:SOURce

---

Description	Sets or reads if the internal or an external clock is use for sample output.	
Syntax	SYSTem:CLOCK:SOURce[?] {INTernal EXTernal}	
Arguments	{INTernal EXTernal}: Specifies whether to use the internal clock (default) or an external 10MHz clock on the clock input connector (same as TRIG-IN 4).	
Query response	Returns the selected currently used sample clock source.	
Example(s)	>SYST:CLOC:SOUR EXT	- Makes the instrument use an externally supplied 10MHz clock for sample output synchronization.
	>SYST:CLOC:SOUR? EXT	- Queries which clock is currently used
Notes	This command is used for synchronizing multiple instruments. Note that the sample output rate is 1MHz, so an externally supplied 10MHz clock will be subdivided. The peak-to-peak amplitude of the signal should be at least 2.2 V and positive. Signals below 5 MHz are ignored. The QDAC-II can also generate a 10MHz clock on the TRIG Out 5 port.	

---

## SYSTem:CLOCK:EXTernal:STATus

---

Description	Sets or reads if the internal or an external clock is use for sample output.	
Syntax	SYSTem:CLOCK:External:STATus?	
Arguments	None	
Query response	Returns 1 (true) if the external 10 MHz clock signal is used, otherwise 0 (false).	
Example(s)	>SYST:CLOC:EXT:STAT?  1	- Check if an externally supplied 10MHz clock for sample output synchronization is being used.
Notes	This command is used for reassuring that an external 10 MHz clock signal (on Trigger In #4) is being used. This is useful, because when setting SYST:CLOC:SOUR to EXT, the external signal is only used if the quality (amplitude) of the signal is acceptable.	

---

## SYSTem:CLOCK:SYNChronize:IMMEDIATE

---

Description	Aligns the (1MHz) sample clock to the next external trigger event on the Sync In port (same as Trigger In #3).	
Syntax	SYSTem:CLOCK:SYNChronize:[IMMEDIATE]	
Arguments	None	
Query response	None. This is an event.	
Example(s)	>SYST:CLOC:SYNC	- Makes the instrument align its 1 MHz output sample clock to the next trigger event on Sync In.

---

**Notes** This command is used for synchronizing multiple instruments sharing the same 10 MHz clock. The :CLOC:SYNC command is basically a “phase alignment” so that the samples output at a 1MHz rate appear simultaneously for all synchronized units.

If there is no 10MHz signal available on the clock input connector, the instrument will use its internal clock. In that case, after SYNC’ing the units may drift apart causing time shifts between the samples out from the different instruments.

When the :CLOC:SYNC command is issued, the instrument will go into *Sync Wait* mode for 2 seconds and wait for a positive flank on the *Sync In* BNC port (*Trigger In #3*). After the two seconds the instrument will go into normal operation mode.

Note that while in *Sync Wait* mode all outputs are paused.

The synchronization pulse is usually received from the *Sync out* port (*Trigger Out 4*) on the master QDAC-II unit.

---

### SYSTem:CLOCK:SEND:ENABLE

---

Description	Enables/disables sending out a 10MHz clock signal on <i>Trigger Out 5</i> .	
Syntax	SYSTem:CLOCK:SEND[:ENABLE][?] {ON OFF}	
Arguments	{ON OFF}:	ON: <i>Trigger Out 5</i> is used to send out the 10MHz signal OFF: <i>Trigger Out 5</i> is used as a normal trigger output
Query response	Returns whether sending out a 10MHz clock on <i>Trigger Out 5</i> is enabled or not.	
Example(s)	>SYST:CLOC:SEND ON	- Makes the instrument send out a 10MHz clock on the clock output port ( <i>Trigger Out 5</i> ) for synchronizing multiple instruments.
	>SYST:CLOC:SEND?	- Queries if <i>Trigger Out 5</i> is currently used to send out a 10MHz clock (ON) or not (OFF)
	1	
Notes	This command is used for synchronizing multiple instruments to the clock of one unit. When CLOCK:SEND is ON, <i>Trigger Out 5</i> is blocked for use as a standard trigger output port. To align the sample output across units a trigger pulse should be sent to the <i>Sync In</i> port ( <i>Trigger In 3</i> ).	

---

### SYSTem:LFRrequency

---

Description	Sets or reads line frequency used by the instrument to convert NPLCycles to APERTure for the current sensor and vice versa.	
Syntax	SYSTem:LFRrequency[?] <numeric_value>	
Arguments	<numeric_value>:	50 or 60. Any other provided value will be converted to the nearest of the two.
Query response	Returns the currently used line frequency for NPLCycles <> APERTure conversion	
Example(s)	>SYST:LFR 50	- Sets the used line frequency to 50 Hz
	>SYST:LFR?	- Queries the currently used line frequency
	50	

---

---

Notes	Used very rarely, primarily at factory or before starting to use the instrument. Because the instrument does not have mains input it cannot faithfully detect the line frequency by itself.
-------	---

---

### SYSTem:REStart

---

Description	Restarts the firmware including LAN interface, and resets all registers and data structures to power up conditions.
Syntax	SYSTem:REStart
Arguments	None
Query response	N/A
Example(s)	>SYST:REST
Notes	Used very rarely, and typically only by the firmware updater.

---



## 7.7 Diagnostic commands

Diagnostic commands are related to service and maintenance of the instrument, and are very specific for the QDAC-II.

### DIAGnostic:VCALibration:{HIGH|LOW}:A

---

Description	Sets or queries the voltage multiplication factor used for transforming a desired voltage to the DAC digital value for the specified channel(s) for the specified voltage range.	
Syntax	DIAGnostic:VCALibration[n]:{HIGH LOW}:A[?] <numeric_value> [,channel_list]	
Arguments	HIGH, LOW:	Specifies the high or low voltage range
	A:	Specifies the multiplication factor in DAC units per Volt.
	MINimum:	1
	MAXium:	1e6
	Default HIGH:	52428.8
	Default LOW:	262144
Query response	Returns the multiplication factor used for converting a desired voltage in units of Volts to DAC units for the specified voltage range.	
Example(s)	<p>&gt;DIAG:VCAL3:LOW:A 476684.5 - Sets the voltage calibration multiplication factor in the LOW voltage range for ch. 3.</p> <p>&gt;DIAG:VCAL3:LOW:A? - Returns the present voltage calibration multiplication factor for the LOW voltage range of ch. 3.</p> <p>4.766845e5</p>	
Notes	<p>Each channel has individual linear calibration constants for its voltage output in both ranges. With these command, new calibration constants can be set, or the existing ones can be read out.</p> <p>The raw DAC digital value is calculated as:  <math>DAC\_value = VOLTage:Level:AMPLitude \times A + B</math></p> <p>In order to persist new calibration parameters between power down and ups, the parameters have to be stored using the CALibration:UPDate command. Note that there is no command for restoring factory settings.</p>	

---

### DIAGnostic:VCALibration:{HIGH|LOW}:B

---

Description	Sets or queries the DAC offset used when transforming a desired voltage to the DAC code for the specified channel(s) for the specified voltage range.	
Syntax	DIAGnostic:VCALibration[n]:{HIGH LOW}:B[?] <numeric_value> [,channel_list]	
Arguments	HIGH, LOW:	Specifies the high or low voltage range
	<numeric_value>:	Integer. Specifies the calibration offset in DAC code units.
	MINimum:	-524288
	MAXimum:	524287
	Default HIGH:	0
	Default LOW:	0
Query response	Returns the DAC offset used for converting a desired voltage in units of Volts to DAC code for the specified voltage range.	

---

---

Example(s)	<pre>&gt;DIAG:VCAL3:LOW:B 231</pre> <p>- Sets the voltage calibration offset voltage in the LOW voltage range for ch. 3.</p> <pre>&gt;DIAG:VCAL3:LOW:B?</pre> <p>- Returns the present voltage calibration offset voltage in the LOW voltage range for ch. 3.</p> <pre>231</pre> <pre>&gt;DIAG:VCAL3:HIGh:A 476684;B 231</pre> <p>- Sets both voltage calibration factors for ch. 3 in the HIGH voltage range</p>
Notes	<p>Each channel has individual linear calibration constants for its voltage output in both ranges. With these command, new calibration constants can be set, or the existing ones can be read out.</p> <p>In order to persist new calibration parameters between power down and ups, the parameters have to be stored using the CALibration:UPDate command. Note that there is no command for restoring factory settings.</p> <p>The raw DAC digital code is calculated as:  <math>DAC\_code = VOLTage:Level:AMPLitude \times A + B</math></p>

---

### DIAGnostic:ICALibration:{HIGH|LOW}:A

---

Description	Sets or queries the current sensor multiplication factor used when transforming a current sensor ADC code to Amperes for the specified channel(s) in the specified current range.
Syntax	DIAGnostic:ICALibration[n]:{HIGH LOW}:A[?] <numeric_value> [,channel_list]
Arguments	<p>HIGH, LOW: Specifies the high or low current range</p> <p>&lt;numeric_value&gt;: Specifies the calibration multiplication factor in units of Ampere/LSB.</p> <p>MINimum: -1</p> <p>MAXium: 1</p> <p>Default HIGH: 2.622605e-9</p> <p>Default LOW: 2.622605e-14</p>
Query response	Returns the multiplication factor used when converting an ADC sensor code to current in units of Ampere/LSB for the specified range.
Example(s)	<pre>&gt;DIAG:ICAL3:HIGh:A 2.62262e-9</pre> <p>- Sets the current calibration multiplication factor in the HIGH voltage range for ch. 3.</p> <pre>&gt;DIAG:ICAL3:HIGh:A?</pre> <p>- Returns the present current calibration multiplication factor in the HIGH voltage range for ch. 3.</p> <pre>2.62262e-9</pre>
Notes	<p>Each channel has individual linear calibration constants for its current sensors in both current ranges. With these command, new calibration constants can be set, or the existing ones can be read out.</p> <p>In order to persist new calibration parameters between power down and ups, the parameters have to be stored using the CALibration:UPDate command. Note that there is no command for restoring factory settings.</p> <p>The measured current in Amperes is calculated as:  <math>Current(Amps) = ADC\_code \times A + B</math></p>

---

## DIAGnostic:ICALibration:{HIGH|LOW}:B

---

Description	Sets or queries the current sensor offset used when transforming a current sensor ADC code to Amperes for the specified channel(s) in the specified current range.	
Syntax	DIAGnostic:ICALibration[n]:{HIGH LOW}:B[?] <numeric_value> [,channel_list]	
Arguments	HIGH, LOW:	Specifies the high or low current range
	<numeric_value>:	Specifies the calibration offset in units of Amperes.
	MINimum:	-1
	MAXimum:	1
	Default HIGH:	0
	Default LOW:	0
Query response	Returns the offset used when converting an ADC sensor digital code to current in units of Ampere for the specified current range.	
Example(s)	<pre>&gt;DIAG:ICAL3:HIGH:B 0.01</pre> <p>- Sets the current calibration offset in the HIGH voltage range for ch. 3.</p> <pre>&gt;DIAG:ICAL3:HIGH:B?</pre> <p>- Returns the present current calibration offset in the HIGH voltage range for ch. 3.</p> <pre>0.01</pre>	
Notes	<p>Each channel has individual linear calibration constants for its current sensors in both current ranges. With these command, new calibration constants can be set, or the existing ones can be read out.</p> <p>In order to persist new calibration parameters between power down and ups, the parameters have to be stored using the CALibration:UPDate command. Note that there is no command for restoring factory settings.</p> <p>The measured current in Amperes is calculated as:</p> $\text{Current(Amps)} = \text{ADC\_code} \times A + B$	

---

## DIAGnostic:CALibration:UPDate

---

Description	Writes the actual DAC and current sensor calibration factors for all channels to the flash memory on the output circuit boards.	
Syntax	DIAGnostic:CALibration:UPDate	
Arguments	None	
Query response	None. This is an event.	
Example(s)	<pre>&gt;DIAG:CAL:UPD</pre> <p>- Writes calibration factors to flash memory.</p>	
Notes	<p>Writing calibration factors to the memory on the output boards is an operation which is usually carried out after calibration and calculation of new calibration factors. This always happens before the instrument leaves the factory.</p>	

---

## DIAGnostic:CCHannel

---

Description	Sets or queries the current sensor offset used when transforming a current sensor ADC code to Amperes for the specified channel(s) in the specified current range.	
Syntax	DIAGnostic:CCHannel <channel_no>	
Arguments	<channel_no>:	Number (0, 1-24) of the channel which should be routed to the CAL output. Channel no zero means that the CAL connector will not be connected to any of the sources.
	Default:	0
Query response	Returns the number of the channel which is currently routed to the CAL output	
Example(s)	>DIAG:CCH 14 >DIAG:CCH?  14	- Connects channel 14 to the CAL output - Returns the number of the channel currently connected to the CAL output.
Notes	The CAL connector on the rear panel is a convenience feature used when calibrating the instrument, as one can connect for example connect a voltmeter to this output and then one by one programmatically connect each channel to this connector and perform the calibration. It is of course also useful for general testing of the device.	

---

7.8 Future section: Reporting of operation status

7.9 Future section: Questionable data (current limit)

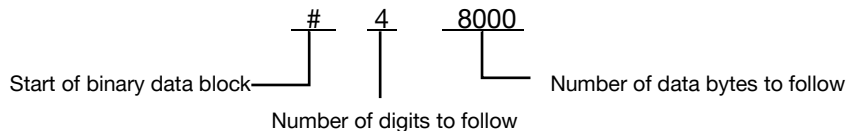
8 Future section: List of error messages

## 9 The IEEE 488.2 binary block format

For commands and queries transferring large amounts of data, there is the option for using a block format. The format is described in IEEE-488.2. For this instrument only a small fraction of the options described in the standard are used.

### Write

For writing data to the instrument, the binary data is preceded by a block header of this format:



The binary block header is followed by the actual data in IEEE 754 single precision floating point format (*binary32*). As an example of 8000 data bytes will give 2000 floating point numbers.

### Read

For reading large chunks of data it is necessary to instruct the instrument to send the data in binary format. This is done on the sub-subsystem (command) level and not as a general setting (which is the intended way by the standard), as only LIST and AWG commands support the binary block format. The IEEE-488.2 standard is implemented in a simplified way using the FORMat sub-system:

```
:FORMat:READings:DATA <type> [,<length>]
```

```
<type>:  ASCII or REAL
```

```
<length>:  32 or 64. Length is only used for REAL.
```

In REAL mode data will be transferred as described above for *write* – in a “IEEE-488.2 definite length block”, that is with the #<digits><bytes> header. In IEEE488.2 default length is 64 bits. This instrument uses 32 bits as default.

## 10 Specifications and performance

Number of channels	24
Voltage ranges	$\pm 10$ V and $\pm 2$ V, configurable by software on each channel individually
Voltage resolution	20 bits: 1 LSB = 3.8 $\mu$ V (2V range), nominally 1 LSB = 19.1 $\mu$ V (10V range), nominally 25 bits DC mode <sup>1)</sup> : 1 LSB = 0.1 $\mu$ V (2V range), nominally 1 LSB = 0.6 $\mu$ V (10V range), nominally
Output resistance	50 Ohms resistive
Output capacitance	1 $\mu$ F / 0.22 $\mu$ F / 10 nF (DC / MED / HIGH filter modes)
Temperature coefficient	Not measured. But data indicates < 5 $\mu$ V/ $^{\circ}$ C @ 23 $^{\circ}$ C, 0.1V output (10V range)
Stability	Less than $\pm 2$ $\mu$ V fluctuations in 14 days (2V range, 1V output, zero current) after minimum 3 hours warm-up (room temperature stable within $\pm 0.5^{\circ}$ C)
Maximum integral non-linearity	1.7 LSB (typical)
Maximum differential non-linearity	0.7 LSB (typical)
Current output, nominally <sup>2)</sup>	$\pm 10$ mA / $\pm 150$ nA (high and low current range respectively)
Sample rate	1 MHz
Rise time (1-99% open)	4 $\mu$ s (in high bandwidth and high current mode))
Predefined waveform generators	Sine, Square, Triangle on every channel.
Arbitrary waveform generator	One on each channel (channels can share waveform definitions)
Traces	16 each with 6 million points (6 seconds) - to be increased in future updates
Trigger outputs (BNC)	3x isolated 5V on front, output resistance 50 $\pm 20$ $\Omega$ 2x non-isolated 3.3V on rear, output resistance not specified but is effectively around 30 $\Omega$
Trigger inputs	4x isolated 3.3V on rear (minimum -0.5V, max 3.8V), input current, typ. +0.01 $\mu$ A, max. $\pm 10$ $\mu$ A. Trigger minimum level $\approx 2$ . V, safe level 2.5V.
Current resolution	Minimum detectable current step: $\pm 10$ mA range: $\approx 10$ $\mu$ A $\pm 150$ nA range: $\approx 50$ pA
Power requirements <sup>3)</sup>	$\pm 15.4$ V (1.5A). Min-max range: $\pm (15.2-15.6)$ V 5.5V (1.5A). Min-max range: 5.2-5.7V
Dimensions (incl. feet)	45 cm x 37 cm x 18.5 cm
Weight	6.4 kg

<sup>1)</sup> 25 bits resolution is available in FIXed DC mode, that is only steady DC output is supported, thus not including sweeps and lists which are still 20 bits.

<sup>2)</sup> The instrument can actually source up to around 18 mA in the high current range on a few channels simultaneously, but not on all. Further, specifications are for  $\pm 10$  mA. In the LOW current range the sourcing  $\pm 10$ V short circuit currents are about +4.3mA and -11 mA, respectively. In LOW current (sense) mode the current sensor is only guaranteed to measure up to  $\pm 150$  nA. However, in most cases it will actually measure up to  $\pm 200$  nA before saturating.

<sup>3)</sup> The requirements are satisfied by the QDevil Power Supply. If for some reason a third-party power supply is required, it is possible to use one when the mentioned requirements are fulfilled.

## 10.1 Noise and drift

### Noise

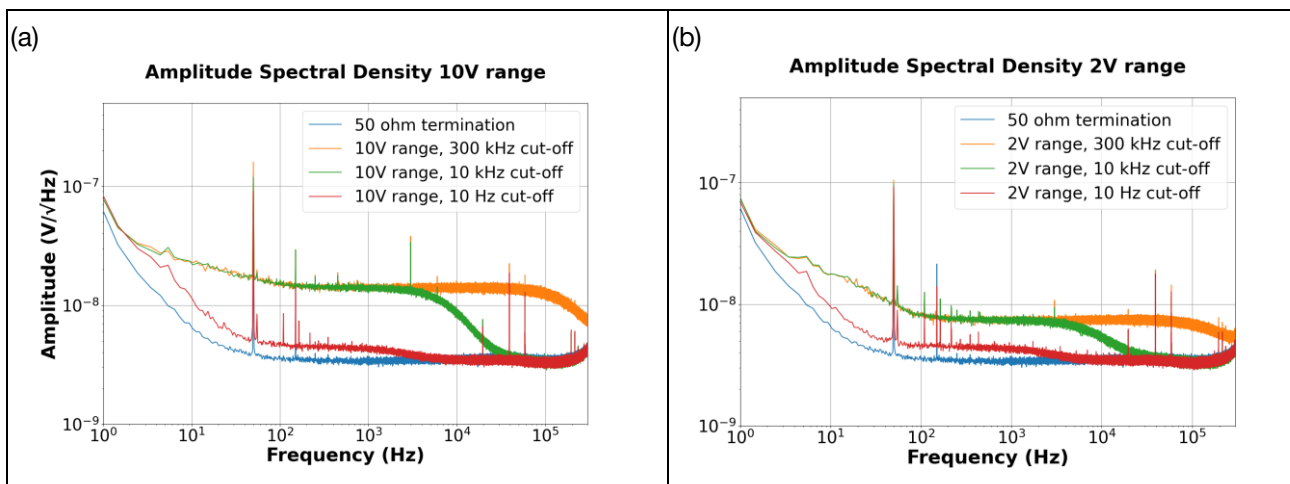
Noise spectra have been derived from multiple time series each of about 34 seconds. The QDAC-II signal was amplified by an AC coupled Stanford Research SR560 low noise pre-amplifier (0.03 Hz high-pass filter) set to a gain of  $10^4$  and a 1 MHz bandwidth. The signal was digitized at 4 MS/s using an AC coupled 8 bit USB oscilloscope (PicoScope 2205A-MSO). The spectra have been binned into approximately 1 Hz wide bins and normalized to power spectral density. 5-10 power spectra were averaged to achieve reasonably smooth curves. To ease interpretation, the power spectra are presented as amplitude spectra. Reference measurements were performed by substituting the QDAC BNC output with a 50 ohm termination. All measurements have been carried out in the QDevil office building *without* access to clean ground, good shielding, temperature control etc. The QDAC-II was powered by a by the QDevil Power Supply (QPS) placed approx. 1.5 m away from the QDAC-II and the SR560.

Measurements were performed both in the 10V range (HIGH) and in the 2V output range (LOW) with the output set to 1V. No load other than that of the SR560 was connected to the QDAC-II output.

The results are shown in Figure 31 and a summarized in Table 6. The  $1/f$  bend is observed around 60 Hz. Above the cut-off frequency of the built-in low pass filters the noise floor settles at the noise level of the setup which is around  $3.5\text{V}/\sqrt{\text{Hz}}$ . The upwards bending of the spectra above 100kHz is an artifact of the measurement setup.

Filter range	10 V range	2V range
HIGH	15 nV/ $\sqrt{\text{Hz}}$	8 nV/ $\sqrt{\text{Hz}}$
MEDium	15 nV/ $\sqrt{\text{Hz}}$	8 nV/ $\sqrt{\text{Hz}}$
LOW	4.5 nV/ $\sqrt{\text{Hz}}$	4.5 nV/ $\sqrt{\text{Hz}}$

Table 6. Noise floor for combinations of filter range and voltage range.



**Figure 31.** Left: Noise spectra in the 10V output range with a DC voltage output of 1V. Curves are shown for the 3 low-pass filter settings, including DC mode which has 25 bits DAC resolution and 10 Hz cut-off. The blue curve shows the background noise measured in a  $50\ \Omega$  termination. Right: The same data measured in the 2V output range, also with a DC voltage output of 1V.



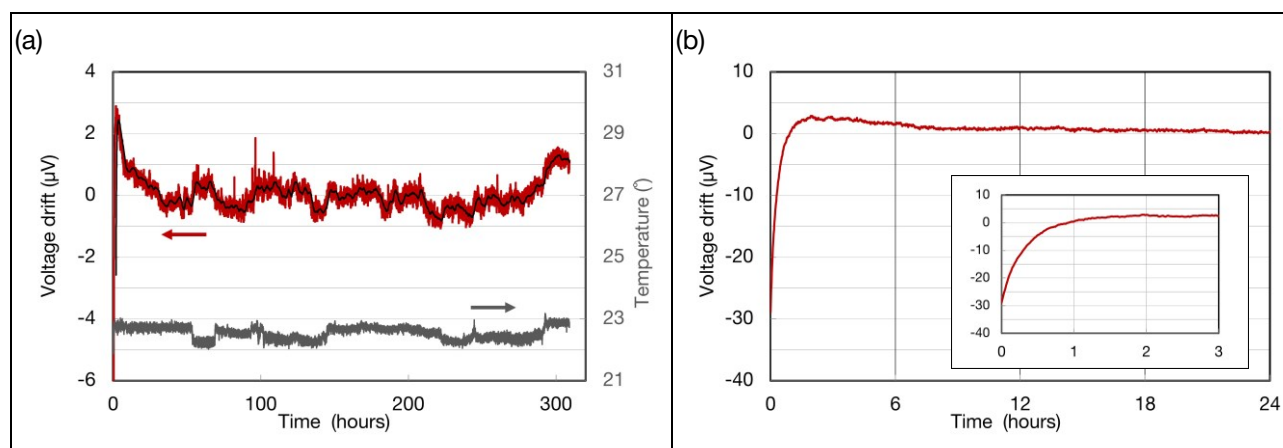
## Drift

Drift is another word for low frequency noise, however studied in the time domain. To measure the low frequency components of a signal, extremely stable equipment is required and full control over the laboratory temperature, or the influence thereof. Therefore, QDevil has engaged with the Danish National Metrology institute (the Danish NIST) for measuring the variation in output when a fixed DC value is set on one of the QDAC-II channels.

After switching on the QDAC-II, the filter mode was immediately switched to DC (without resolution enhancement active), the output range was set to LOW (2V) and the actual output (CH04) to around 1V. The difference between a traceable Zener reference (Fluke 732) and the QDAC-II output was then logged using a high accuracy 8.5-digit DMM (Solartron 7081) for a period of about 12 days. The voltage was averaged and logged in 1 minute time intervals. The QDAC-II channel was only loaded by the input of the DMM.

The temperature in the laboratory was kept stable at nominally  $23.0 \pm 0.5$  °C. The temperature 20 cm from the QDAC-II was logged during the measurement. The laboratory was not free from people traffic, but a small room divider was placed next to the table with the setup in order to shield any draft coming from people passing by. So, a quite realistic experimental situation.

A summary of the measurements can be seen in Figure 32. After an initial relaxation period of about 3 hours the output is stable to within about  $\pm 2$   $\mu\text{V}$  at low voltages and currents if the ambient temperature, as here, is kept stable within  $\pm 0.5$  °C. As most drift from experience is due to the internal voltage reference, it is expected that the drift is approximately the same in the 10V (HIGH) output range. Some correlation between the close-by ambient temperature and the QDAC-II output voltage can be observed, however, the correlation is a bit ambiguous.



**Figure 32.** (a) Change in output voltage recorded from 2-3 minutes after power-on at 1V output in DC mode in the 2V range. Recorded by the Danish National Metrology institute, DFM, using a traceable Zener reference and a precision nano voltmeter. The voltmeter has an integration time of about 1 minute, so the highest frequency is around 30 mHz. The temperature was logged close to the instrument and can be seen to be around  $22.5^{\circ}\text{C} \pm 0.5^{\circ}$ .

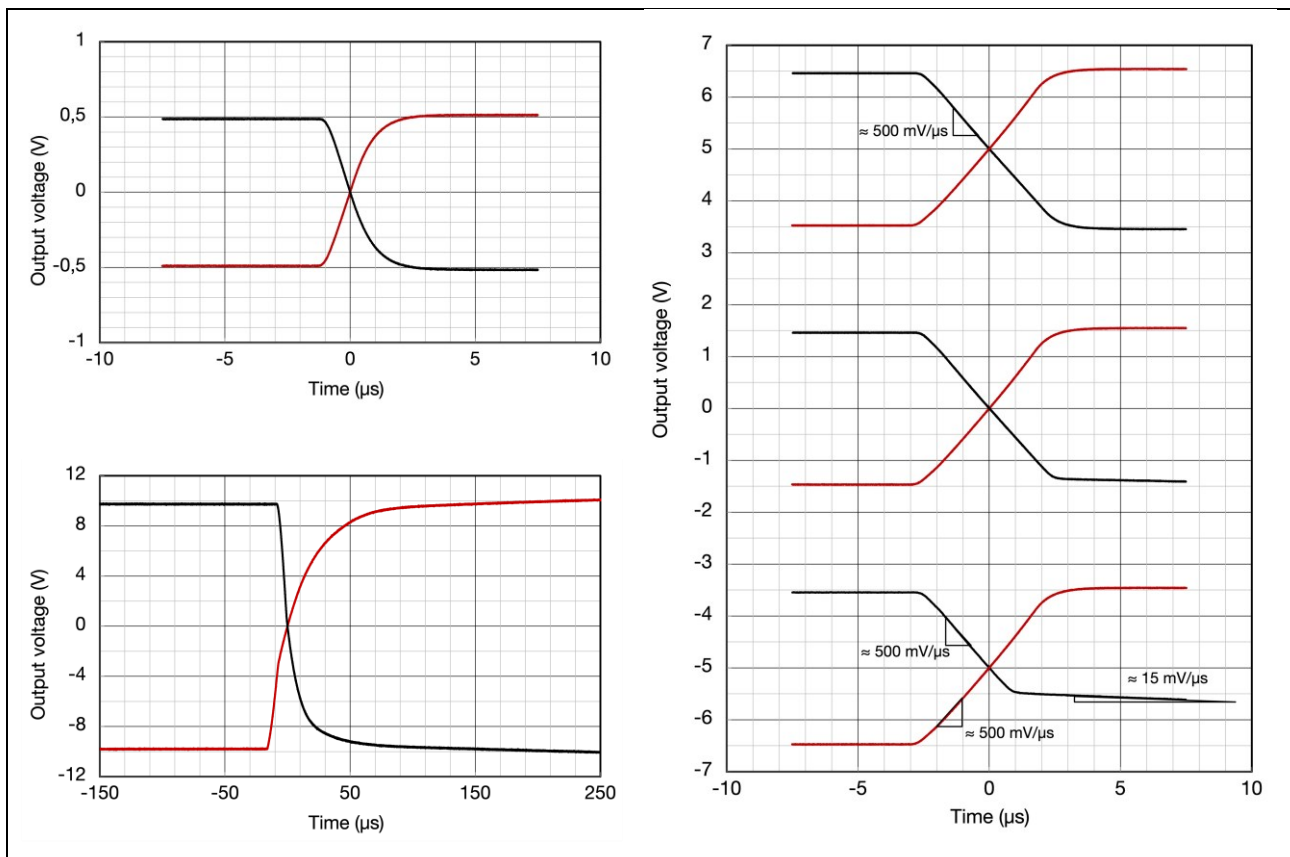
(b) The first 24 hours of drift after power on. It is seen that after about 3 hours the output is close to its final value. But allowing a warm-up of 24 hours will settle the output to vary less than  $\pm 2$   $\mu\text{V}$ , approximately.

## 10.2 Non-linear output characteristics

### 10.2.1 LOW current mode

The nonlinearity of the IV convertor used for current sensing causes the frequency response of the voltage generators in the instrument to be non-linear too. This is primarily in the LOW current mode, and in particular for negative going fast signals. Effectively the bandwidth becomes limited the closer one approaches the negative voltage limit (-10 V) where a slew rate limit of the order of 15 mV/V. This corresponds to the maximum slope of a 2Vpp sine at 2.4KHz.

*Note that in the HIGH current mode these limits are **not** observed.*

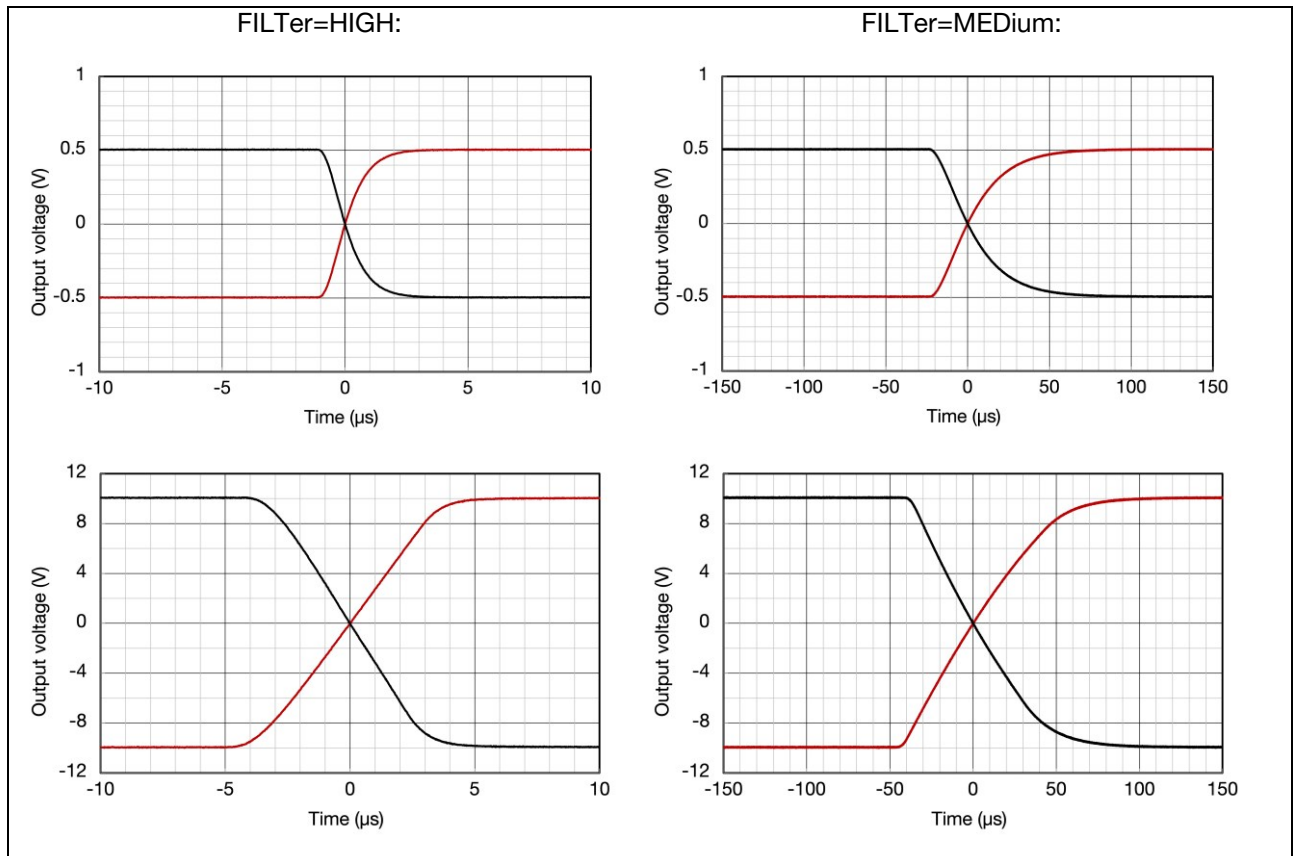


**Figure 33.** Step responses in CURRent:RANGe = LOW mode.

**Left:** Small step and large step symmetrically around zero **HIGH filter mode** (300 KHz cut-off) in the LOW current mode with no external load connected to the channel. It is seen that for low voltages (e.g. 1 V) a  $\approx 99\%$  rise time of 3-4  $\mu\text{s}$  is achieved whereas for a 20V step it is more like 400  $\mu\text{s}$  with an initial steep slope but falling off a few volts.

**Right:** 3 V steps recorded at different DC offsets also in **HIGH filter mode**. It is seen that the response is slew rate limited at around 500 mV/ $\mu\text{s}$  for positive going steps and for negative going steps starting at high voltages. But at negative going steps larger than  $\approx 2\text{V}$  the slew rate settles at around 15 mV/ $\mu\text{s}$  for negative voltages.

## 10.2.2 HIGH current mode



**Figure 34.** Step responses in CURRent:RANGe = HIGH mode with no external load connected to the channel

**Left:** Small step and large step symmetrically around zero in **HIGH filter mode** ( $\approx 300$  kHz cut-off) in the **HIGH current mode**. It is seen that for low voltages (e.g. 1 V) an *RC filter like*  $\approx 99\%$  rise time of 3-4  $\mu\text{s}$ , is achieved whereas for a 20V step it is more like 8  $\mu\text{s}$  partially limited by the output amplifier slew rate.

**Right:** Small step and large step symmetrically around zero in **MEDIUM filter mode** ( $\approx 10$  kHz cut-off) in the **HIGH current mode**. Both for low and high steps an *RC filter like* response is observed with a  $\approx 99\%$  rise time of  $\approx 80$   $\mu\text{s}$  for small steps and 120  $\mu\text{s}$  for the 20V step, slightly limited by the output amplifier slew rate.

### 10.3 Filter switch transients

When switching almost anything it is unavoidable that some transient signal will appear somewhere. That is also true when switching the low-pass filters in the QDAC-II. However, a special circuit (patent pending) developed for avoiding transients ensures that there are no noticeable, or vanishing, voltage steps occurring at the output when switching the filters. However, some remaining transients induced by the relays remain. There are *step signatures* and *ringing signatures* present, depending on the actual transition (HIGH  $\rightarrow$  LOW, HIGH  $\rightarrow$  MED, etc.).

Figure 35 shows examples of the characteristic transient signatures, in this case from a HIGH $\rightarrow$ DC filter switch with a DC output at +9.9 V. For the measurements the QDAC-II output was AC coupled (via an RC high-pass filter, cutoff  $\approx$ 1 kHz) to a cascaded SR445 amplifier. The measurements were performed at a 350 MHz bandwidth but sampled at 60 MHz by an oscilloscope.

In Figure 35(a) it is seen that the voltage step at the switch is smaller than 100 $\mu$ V. The noise which can be observed both before and after the switch is primarily from the cascaded SR445 and not from the QDAC-II itself. In Figure 35(b) an example of a *ringing signature* is shown, actually one of the clearest ones measured. These are characterized by a short ( $\approx$ 0.1 $\mu$ s) very high frequency oscillation and a slow ( $\approx$ 0.5 $\mu$ s) decaying oscillation at around 10 MHz.

For normal use cases low pass filtering will be present in the signal chain, e.g. a QFilter. This will effectively take care of the high frequency transient signals. In the curves below, the grey lines show the data filtered by a digital implementation of an RC filter with a 1MHz cut-off frequency.

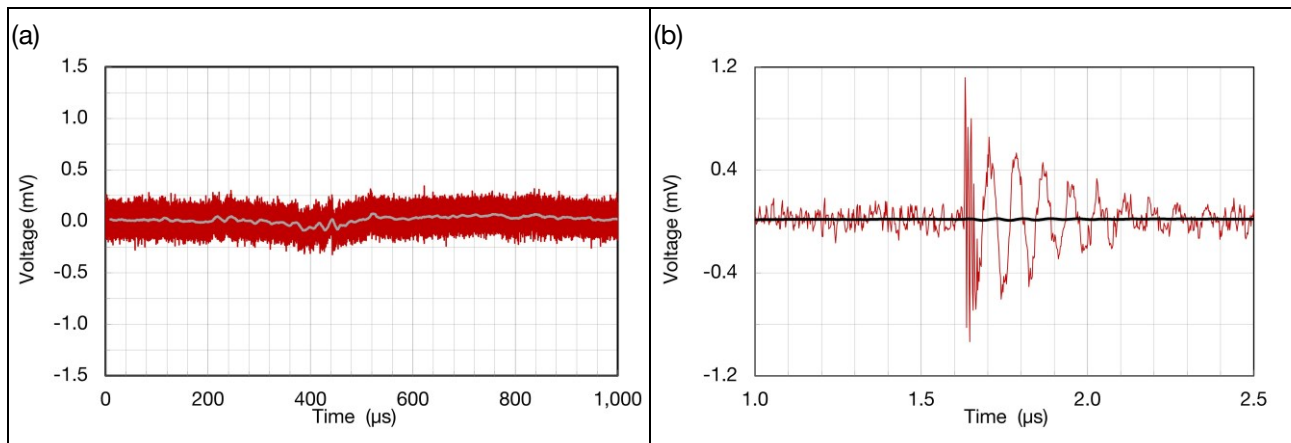
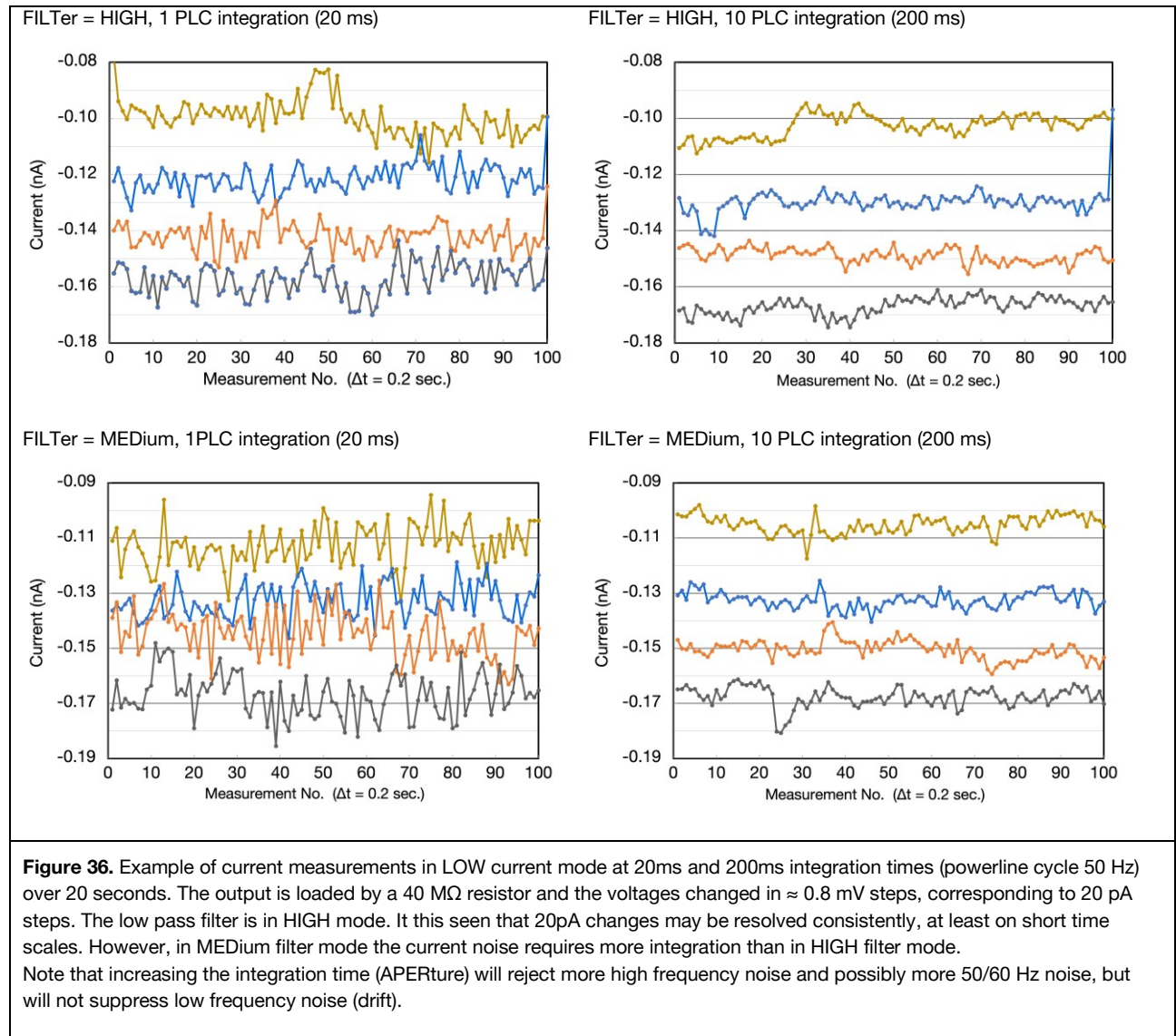


Figure 35. (a) The actual remaining step (change) in voltage when switching from HIGH to DC filter mode at an output level of +9.9 V. (b) One of the clearest (worst) examples of the characteristic *ringing signature* observed during switching from HIGH to DC mode.

The red lines are the recorded data. The grey lines are after filtering the signals with a digital implementation of an RC filter with a 1MHz cut-off frequency.

## 10.4 Current measurement signal to noise ratio

The current sensor's signal to noise ratio reading depends on the magnitude of the current and on the integration time (APERture), but in particular on the used lowpass FILTER (DC, MEDIUM, HIGH, as suppression of voltage noise at the output RC stages is converted to current noise). The following graphs show series of current readings recorded over 20 seconds using different filters and integration times. The channel was loaded by a 40 M $\Omega$  resistor. For each trace the voltage was stepped by  $\approx 0.8$  mV corresponding to an offset of current of nominally 20 pA.



## 11 Bibliography

IEEE Standards Board. (1992). *IEEE Standard Codes, Formats, Protocols, and Common Commands for Use With IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation*. New York, USA: The Institute of Electrical and Electronics Engineers, Inc.

SCPI Consortium. (1999). *Standard Commands for Programmable Instruments (SCPI)*. Retrieved from <https://ivifoundation.org/scpi>

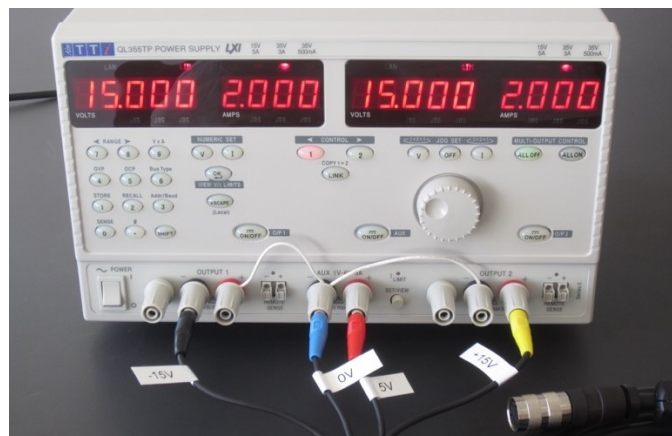
## Appendix A Using laboratory power supplies

Though it *is* recommended to use the QDevil Power Supply for powering the QDAC-II it is also possible to use a third party power supply of good quality, which can switch all voltages on simultaneously.

### Appendix A1. Procedure for configuring a QL355TP power supply

1. Before starting, ensure that the QDAC-II 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  $\sim 1\text{mm}^2$  (not included).
3. Set Output 1 and Output 2 to 15.5V and a 2.00A current limit.
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-II power cord to the power supply outlets as shown below. Note the voltages printed on the flags on the wires:

Black		-15.5 V	OUTPUT1 -
Blue		0 V (Common)	AUX -
Red		+5.5 V	AUX +
Yellow		+ 15.5 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-II 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.

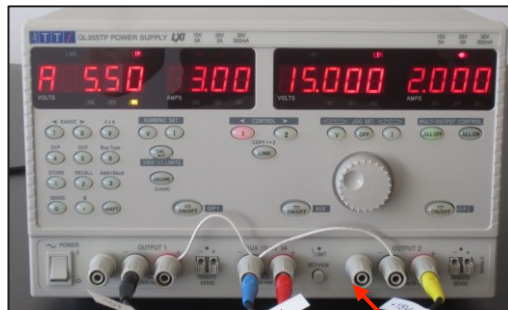
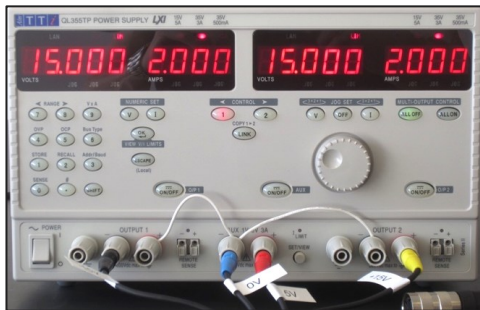
If you have difficulties setting up the power supply or the QDAC-II unit, or if you have other questions, please contact QDevil at [info@qdevil.com](mailto:info@qdevil.com).

## Appendix A2. Procedure for turning ON the QDAC-II using a QL355TP

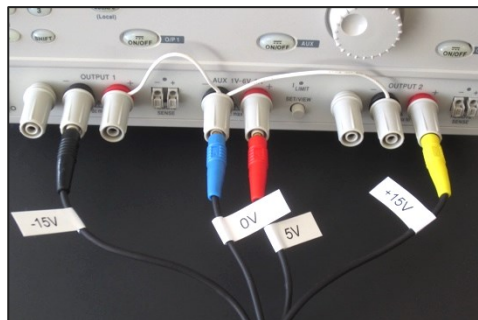
### WARNING

The QDAC-II 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-II. 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).
9. Notice that the LED on the front panel of the QDAC-II 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-II unit, press the **ALL OFF** button on the power supply **before** disconnecting the plug from the back side of the QDAC-II.

Please wait minimum 5 seconds after switching off the instrument until you switch it on again.



## Appendix B Open-source licenses

This is a list of open-source licenses used in the firmware.

Xilinx firmware libraries (SPDX: MIT):

```

/*****
 *
 * Copyright (C) 2007 - 2018 Xilinx, Inc. All rights reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * XILINX BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
 * OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * Except as contained in this notice, the name of the Xilinx shall not be used
 * in advertising or otherwise to promote the sale, use or other dealings in
 * this Software without prior written authorization from Xilinx.
 *****/

/* print.c -- print a string on the output device.
 *
 * Copyright (c) 1995 Cygnus Support
 *
 * The authors hereby grant permission to use, copy, modify, distribute,
 * and license this software and its documentation for any purpose, provided
 * that existing copyright notices are retained in all copies and that this
 * notice is included verbatim in any distributions. No written agreement,
 * license, or royalty fee is required for any of the authorized uses.
 * Modifications to this software may be copyrighted by their authors
 * and need not follow the licensing terms described here, provided that
 * the new terms are clearly indicated on the first page of each file where
 * they apply.
 */

```

LwIP TCP/IP stack (SPDX: BSD-3-Clause):

```

/*
 * Copyright (c) 2001-2004 Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 * derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
 * SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 * OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 *
 * This file is part of the lwIP TCP/IP stack.
 *
 * Author: Adam Dunkels <adam@sics.se>
 * Author: Simon Goldschmidt
 * Improved by Marc Boucher <marc@mbsi.ca> and David Haas <dhaas@alum.rpi.edu>
 */

/*
 * Copyright (c) 2001-2004 Leon Woestenberg <leon.woestenberg@gmx.net>
 * Copyright (c) 2001-2004 Axon Digital Design B.V., The Netherlands.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 * derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

```

\* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT  
\* SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,  
\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT  
\* OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS  
\* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN  
\* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING  
\* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY  
\* OF SUCH DAMAGE.  
\*  
\* This file is part of the lwIP TCP/IP stack.  
\* The Swedish Institute of Computer Science and Adam Dunkels  
\* are specifically granted permission to redistribute this  
\* source code.  
\*  
\* Author: Leon woestenberg <leon.woestenberg@gmx.net>  
\*/

---