octoDAC performance test report

Rusty Nicovich

May 19, 2020

1 Motivation

A gap exists in the market for a low-cost USB-controlled digital-to-analog converter (DAC) with at least six independent outputs. These outputs should cover a 0-5v range natively, with the option for a different range for specific applications. High timing resolution, low jitter, and high precision on the output is nice but not a requirement.

The primary use case this unit would be used for is programmatic control of laser output through an analog input on the laser driver. There even 8 bit control (approximately 0.5% per step) would be plenty of output precision. A simple use case exists where sub-second updates to the value of an otherwise steady output is sufficient. A slightly more sophisticated use case would use simple analog waveforms with some reasonable (sub-millisecond) timing resolution. These downstream devices would be part of a fluorescence microscope where data acquisition times are nearly always on the timescale of tens to hundreds of milliseconds.

The octoDAC shield and sketch was designed with these use cases in mind. An Arduino Uno provides the USB communication and uses SPI to communicate with a Texas Instruments DAC8568A octal DAC. The 0-2v5 outputs from this IC passes through a pair of quad op amps, each set up as a gain = 2 buffer. These outputs feed into SMA connectors to outside devices.

For testing the octoDAC board was stacked between an Arduino Uno and NicoLase board. A second NicoLase board on an Arduino Uno serves as the trigger device. A 3 ms pulse is sent from the triggering NicoLase output following a USB serial command. This pulse is typically used to trigger the oscilloscope and the MASTERFIRE pin on the receiving NicoLase + octoDAC combo. Output and trigger signals are measured on a Tektronix TDS 2012C dual-channel oscilloscope. All measurements are given as mean ± standard deviation of 100 trials unless otherwise noted.

2 Timing performance

2.1 Digital timing precision

The receiving NicoLase is configured to output on all channels upon receiving a trigger signal to its MASTERFIRE pin. The average time delay between the trigger pulse and output pulse is $5.15\,\mu sec \pm 0.166\,\mu sec$ (Figure 1).

This time is effectively the instrument response rate of the device. The incoming digital trigger is driven by an interrupt through simple function to drive the output pin high. For the designed application of the NicoLase this is more than sufficient response speed. It also serves as a convenient benchmark for the timing performance of the octoDAC.

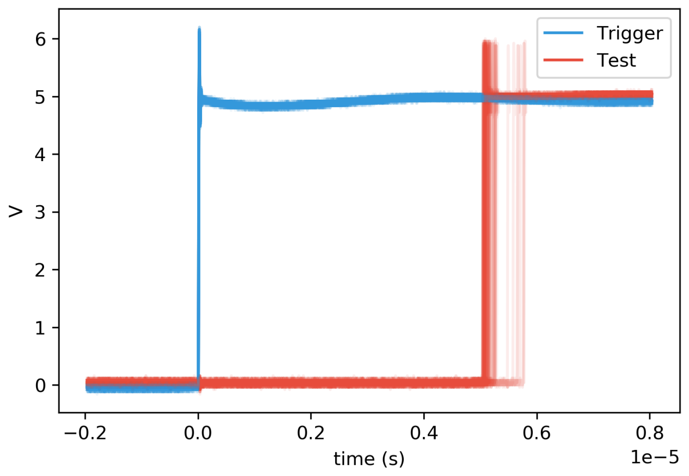

Figure 1: Digital timing precision. Delay between trigger pulse and output digital pulse is $5.15\,\mu sec \pm 0.166\,\mu sec$.

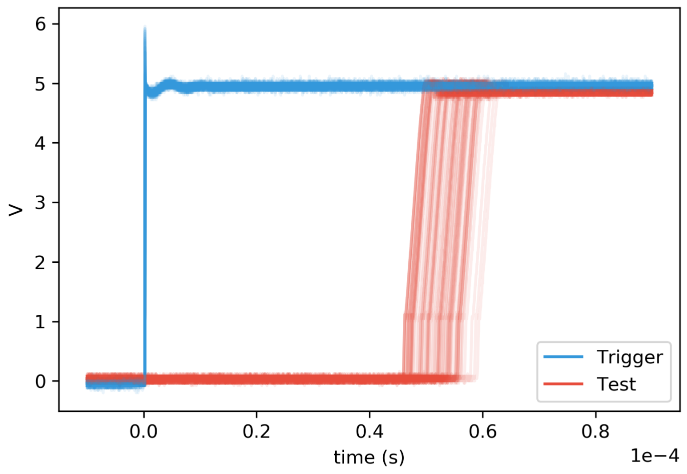

Figure 2: Analog timing precision. Delay between trigger pulse and output analog pulse on a single DAC channel is $52.36\,\mu sec \pm 3.49\,\mu sec$.

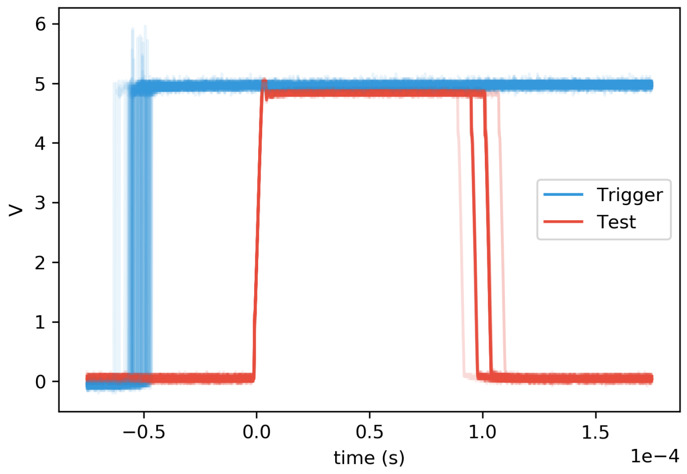

Figure 3: Single waveform timing precision. Deviation from a nominally 100 μsec pulse on a single DAC channel is measured as $0.134\,\mu sec \pm 3.38\,\mu sec$.

2.2 Analog timing precision

The receiving NicoLase triggers the output of a top hat pulse on the receipt of an incoming pulse to the MASTERFIRE pin. The measurement quantifies the time delay between the trigger pulse and the rise of this top hat signal from the analog output on a single DAC channel. The delay is measured as $52.36\,\mu sec \pm 3.49\,\mu sec$ (Figure 2).

The analog signal is driven by the main loop of the Arduino and not an interrupt pin. The DAC signal also requires a transfer over the SPI bus to execute. It is not surprising that the delay is approximately 10x longer for this analog signal compared to the digital signal. This timing still represents a timescale much faster than nearly all devices on a standard fluorescence microscope.

2.3 Single waveform timing precision

The rise time of a top hat pulse triggers the oscilloscope. A single channel of the DAC outputs a pulse nominally 100 μsec long before returning to ground. The measurement quantifies the error from this nominal pulse width. The error in this pulse is measured as $0.134\,\mu sec \pm 3.38\,\mu sec$ (Figure 3).

This timing performance is on a single DAC channel and within a continuous waveform. This represents the ideal case for this device. Under these conditions timing performance is well beyond any requirements for the stated use case.

2.4 Analog cycle timing precision

The octoDAC is set to output a free-running square wave with nominal 100 μsec between the falling and rising edges of the waveform. This portion of the waveform is set to be the end of one cycle of the Arduino main loop and the start of the next. Or, the waveform waypoints includes a 100 μsec delay after the falling edge. Additional delay would be due to the overhead of the Arduino loop to start the next rise time. This additional delay is measured as $63.0\,\mu sec \pm 3.25\,\mu sec$.

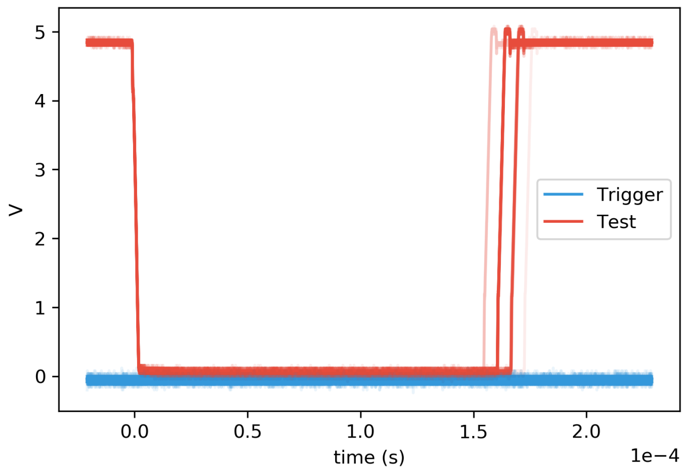

Figure 4: Analog cycle timing precision. Deviation from a nominally 100 μsec pulse on a single DAC channel is measured as $63.0\,\mu sec \pm 3.25\,\mu sec$.

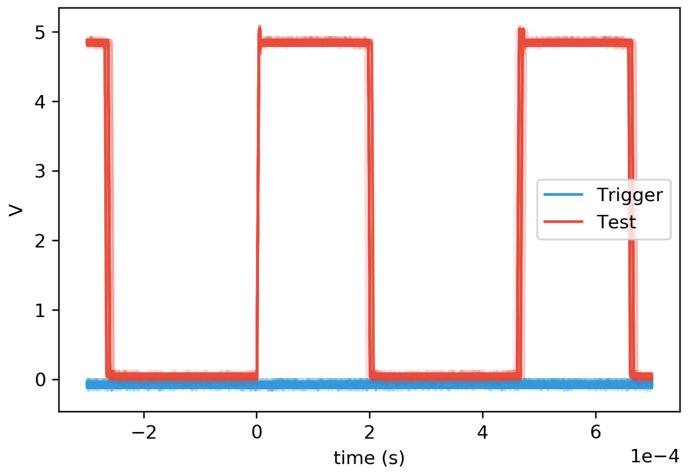

Figure 5: Waveform timing with naive inputs. A 2.5 kHz square wave is made from a rising edge, hold, falling edge, and hold before repeating the cycle in free-running mode. Deviation from a nominally 200 μsec high period is measured as $1.63\,\mu sec \pm 3.11\,\mu sec$ and from a nominally 200 μsec low period is measured as $64.8\,\mu sec \pm 3.08\,\mu sec$.

This delay is a small additional delay on top of the analog jitter. Compared to the timing within a waveform, measured as the top hat jitter, the delay here is applied to the start of a waveform. This is relevant when a periodic waveform with predictable cycle time and duty cycle is desired.

2.5 Waveform timing - naive inputs

The octoDAC is set to output a free-running square wave of 2.5 kHz and 50% duty cycle on a single DAC channel. This corresponds to a nominal $200\,\mu sec$ time for each the high and low states of the square wave. The waveform is encoded as a rising edge at $t = 0$, falling edge at $t = 200\,\mu sec$, and hold at the low value for an additional $t = 200\,\mu sec$. After this delay the cycle repeats, as described in 2.4. Deviation from the expected time period is quantified for both the high and low portions of the square wave. Deviation of the high portion of the waveform is measured as $-1.63\,\mu sec \pm 3.11\,\mu sec$. Deviation of the low portion of the waveform is measured as $64.8\,\mu sec \pm 3.08\,\mu sec$ (Figure 5).

The deviations from the nominal periods in the programmed waveform correspond to those values measured in 2.3 and 2.4. Timing precision within a waveform is very high, measured is single $musec$ of deviation. Between waveform cycles the deviation is approximately 20-fold larger. However, the jitter on this delay is small at a few μsec.

2.6 Waveform timing - adjusted inputs

The same waveform is set as in 2.5. However, the delay hold after the falling edge is reduced by $62\,\mu sec$ to compensate for the additional delay introduced at the start of each waveform cycle. The deviation from nominal $200\,\mu sec$ high and low periods is measured as $-2.81\,\mu sec \pm 1.69\,\mu sec$ for the high period and $1.04\,\mu sec \pm 3.08\,\mu sec$ for the low period (Figure 6).

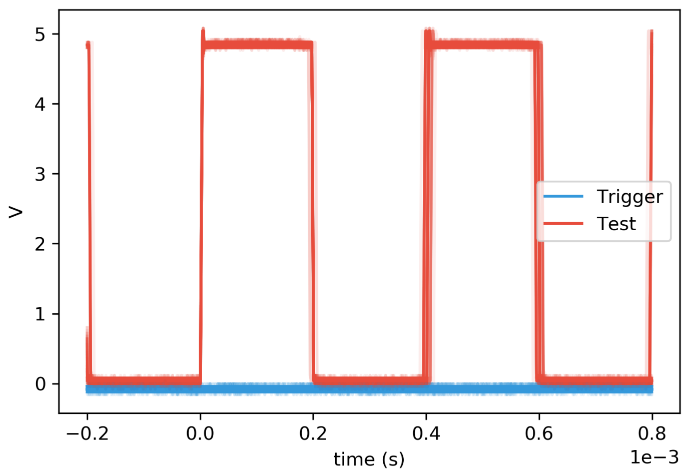

Figure 6: Waveform timing with corrected inputs. A 2.5 kHz square wave is made from a rising edge, hold, falling edge, and hold before repeating the cycle in free-running mode. The terminal hold time is reduced by 62 μsec from the nominal value. Deviation from a nominally 200 μsec high period is measured as $-2.81\,\mu sec \pm 1.69\,\mu sec$ and from a nominally 200 μsec low period is measured as $1.04\,\mu sec \pm 3.08\,\mu sec$.

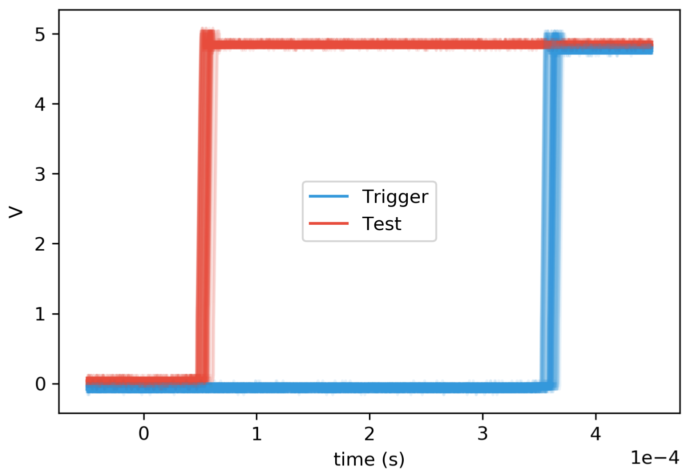

Figure 7: Analog output delay after trigger measured on first and last channels when all channels are activated at $t = 0$. The delay between the incoming trigger and first channel is measured as $54.2\,\mu sec \pm 3.68\,\mu sec$; for the last channel this value is measured as $361.1\,\mu sec \pm 4.18\,\mu sec$.

With this empirical correction for additional delay between cycles, a precise cycle time is realized. Waveforms of > kHz rates can be achieved once this correction is applied.

2.7 Analog delay on all channels

The octoDAC default configuration sets a value on each channel sequentially. A delay between an incoming trigger and rising edge of an analog output is measured as in 2.4. Previously this value was measured for a single channel only. Here values are set for all eight output channels at $t = 0$. The delay between the incoming trigger and analog output waveform is measured for the first and last channels specified in the waveform array. The delay between the incoming trigger and first channel is measured as $54.2\,\mu sec \pm 3.68\,\mu sec$; for the last channel this value is measured as $361.1\,\mu sec \pm 4.18\,\mu sec$ (Figure 7).

The delay between the trigger input and last channel is approximately seven times greater than that between the trigger and first channel. This confirms the nature of this delay is due to the communication overhead between the Arduino and DAC device through the SPI bus. Were this communication time to increase, the delay could decrease. The SPI bus is capable of much higher communication speeds than supported on an Arduino Uno, suggesting a different controller device could be used if analog output with less lag is required.

The octoDAC device here can be configured to operate in asynchronous update mode. There each channel value is loaded into the register but not applied at the output until the LDAC pin on the DAC8568 is toggled. This mode can be supported with the proper jumpers are applied and configured in the Arduino sketch. This exercise is left to the reader.

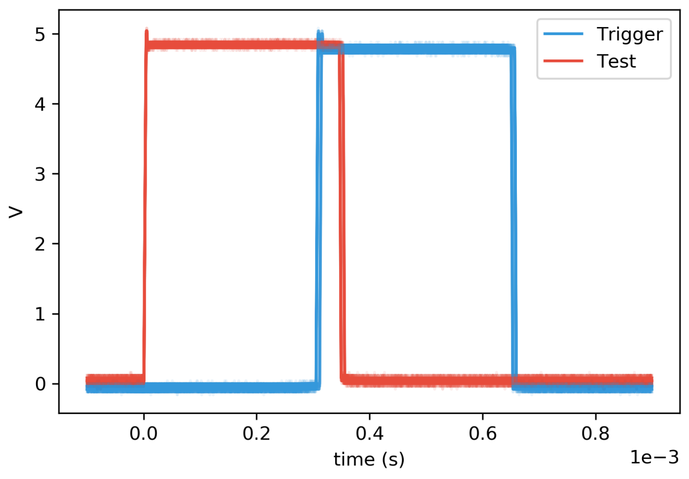

Figure 8: Pulse width for programmed waveforms measured on first and last channels when all channels are activated at $t = 0$. Deviation from the nominal $100\,\mu sec$ pulse width is measured as $246.7\,\mu sec \pm 2.85\,\mu sec$ for the first channel and $247.7\,\mu sec \pm 2.91\,\mu sec$ for the last channel.

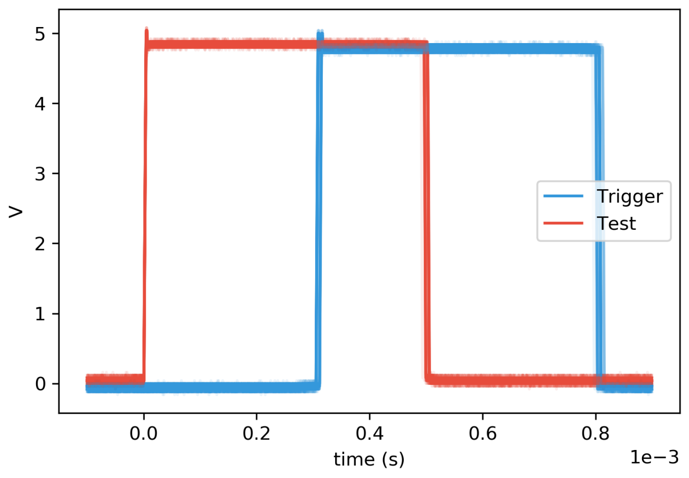

Figure 9: Pulse width for programmed waveforms measured on first and last channels when all channels are activated at $t = 0$. Deviation from the nominal $500\,\mu sec$ pulse width is measured as $-2.66\,\mu sec \pm 4.80\,\mu sec$ for the first channel and $-7.6\,\mu sec \pm 3.05\,\mu sec$ for the last channel.

2.8 Single waveform timing on all channels

The test configured in 2.3 is repeated with signals on all channels, in parallel to the test performed in 2.7. Here a top hat waveform is programmed for all channels and the pulse width of this wave measured on each. The nominal pulse width is $100\,\mu sec$. Deviation from this nominal value is measured as $246.7\,\mu sec \pm 2.85\,\mu sec$ for the first channel and $247.7\,\mu sec \pm 2.91\,\mu sec$ for the last channel (Figure 8).

This deviation is larger than expected. The deviation is also approximately equal to the delay between the rise times of the first and last channels as measured in 2.7. This implies a minimum time is required for the timing ability of the Arduino to be realized. Otherwise the pulse width of the waveform will be defined by the cumulative delay introduced by the SPI communication.

To confirm, the same test was performed with a nominal pulse width of $500\,\mu sec$. This nominal pulse width is greater than the cumulative delay for communicating with eight DAC channels. With a $500\,\mu sec$ nominal pulse width, the deviation is measured as $-2.66\,\mu sec \pm 4.80\,\mu sec$ for the first channel and $-7.6\,\mu sec \pm 3.05\,\mu sec$ for the last channel. This demonstrates the precision of timing during a waveform, but only if the Arduino timing can dictate waveform events. It should be noted that these two waveforms are of precise width, but approximately $350\,\mu sec$ delayed. If alignment of waveforms with more precision than this affords, the asynchronous update described in 2.7 could be a solution.

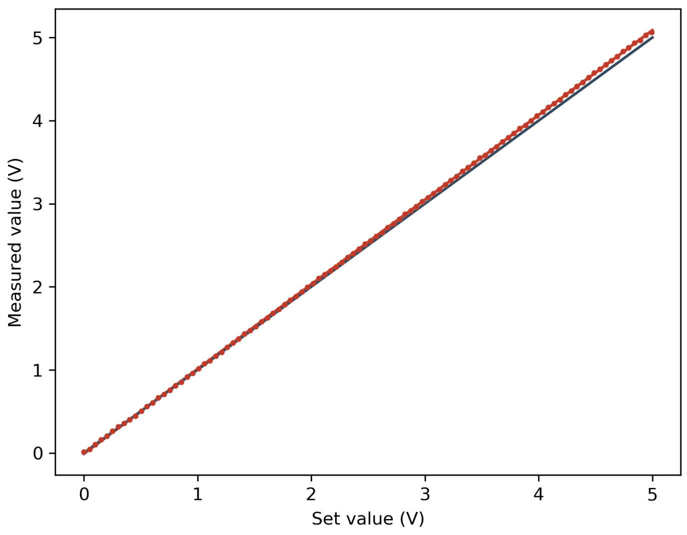

Figure 10: Linearity of voltage output relative to input values. Results for all 8 channels are highly similar and plotted here. The gray dashed line indicates slope = 1.

3 Voltage performance

3.1 Linearity

For each channel, 100 values are set between 0 and 65535. These values should nominally correspond to 0 and 5V output, with equally-spaced steps in between. Each value is held for $250 msec$. The mean value at each point is plotted against the expected value and this plot fit to a line. These fits yield a slope of 1.01905 to 1.01966 and offset of -0.00966 to -0.01078. Measured value and fits are shown in Figure 10.

These results show a response that is highly linear across the output range. The small offset and deviation from slope = 1 could be a factor of measurement error. Even if not, the use case is for systems where relative output is more critical than absolute voltage precision. These deviations from ideal behavior are easily tolerated.

3.2 Voltage resolution

For a single channel, 165 steps of 8 units each are taken between values of 32112 and 33432, corresponding to nominal values of 2.45 V to 2.55 V. A step of 8 units corresponds to a nominal step of 0.61 mV.

Each value is held for 0.5 sec. The mean and standard deviation of each step value is plotted against its nominal value with error bars corresponding to the standard deviation of the measured voltage over the measurement window. Results appear in Figure 11.

The measured values over this range are nearly monotonic, though some deviation from linear behavior is observed. The standard deviation of the measured voltage in this time window ranges from 6.58 mV to 20.12 mV, with a mean of 16.2 mV. These residual voltage ripples correspond to 6.43 to 8.04 bits of resolution. There remains 8-10 bits of voltage resolution without relying on any averaging or filtering. This would be a worst-case scenario and most use cases for this device would act as a low-pass filter for much higher achievable voltage resolution. In such a scenario the resolvable values would be closer to the mean values presented here, with clear discrimination between points separated by 2-3 steps (3-4 bits). This leaves approximately 12 bits of useful voltage range.

3.3 Stability across voltage range

The measured voltage ripple increases at center values near the top of the voltage range Figure 12 shows ripple measured at 0.54 V and 5 V output. Deviations from normal behavior are seen at a value near the upper range of the device.

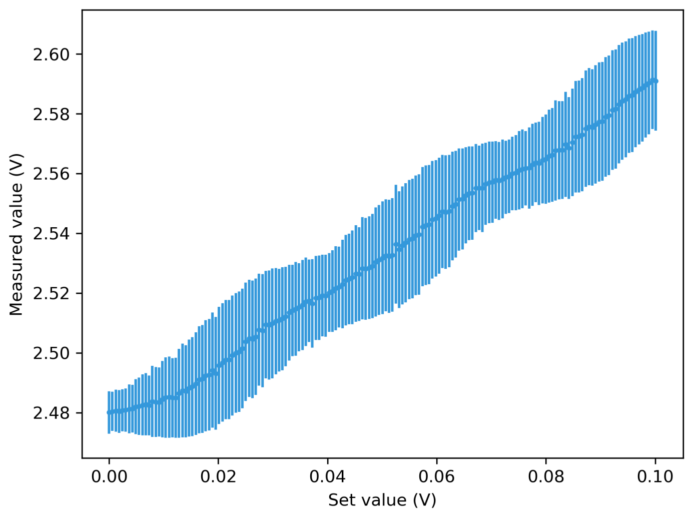

Figure 11: Voltage output plotted as a function of nominal value for 165 values between 2.45 V and 2.55 V. Error bars indicate standard deviation of measured voltage over a 0.5 sec window, yielding values of 6.58 mV to 20.12 mV.

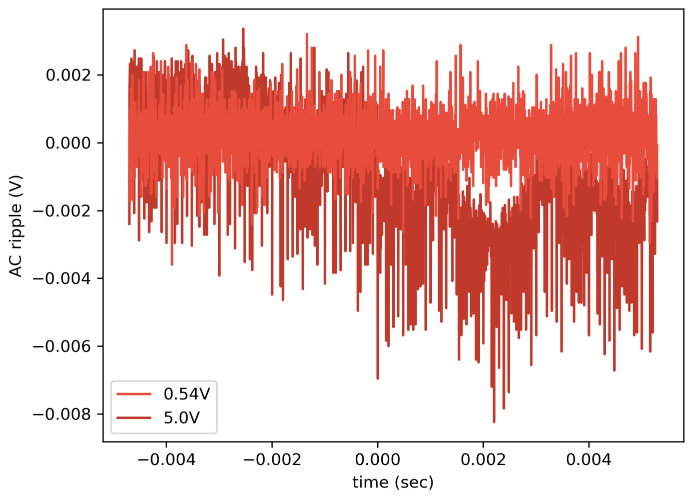

Figure 12: AC voltage stability at indicated DC voltages for a single channel. Values become less stable when near the limit of the output range.