5/23/2024

Sig_Gen V1.1

A simple cost-effective demo/test/training board for an oscilloscope

Paul Wasserman
pwproj24@gmail.com

Contents

Introduction 2

Creation 2

Commands 3

Functionality 4

10/*20* MHz 50% Duty Cycle Waveform 5

100/*200* KHz 50%, 25%, 10%, 5%, 1% Duty Cycle Waveforms 6

1/*2* KHz 50% Duty Cycle Waveform 8

60/*120* MHz 50% Duty Cycle Waveform 9

UART Waveform 11

Manchester Encoded Data Waveform 12

Low frequency PWM - DAC Waveforms 13

Ramp 13

Sawtooth 14

Staircase 15

Sine 16

Full Wave Rectified Sine 17

Gaussian Staircase 18

Uniform Staircase 19

Frequency Settable Sine 20

I2C Waveforms 21

SPI Waveforms 23

Measurement Techniques 24

Custom Measurement Jig Photos 25

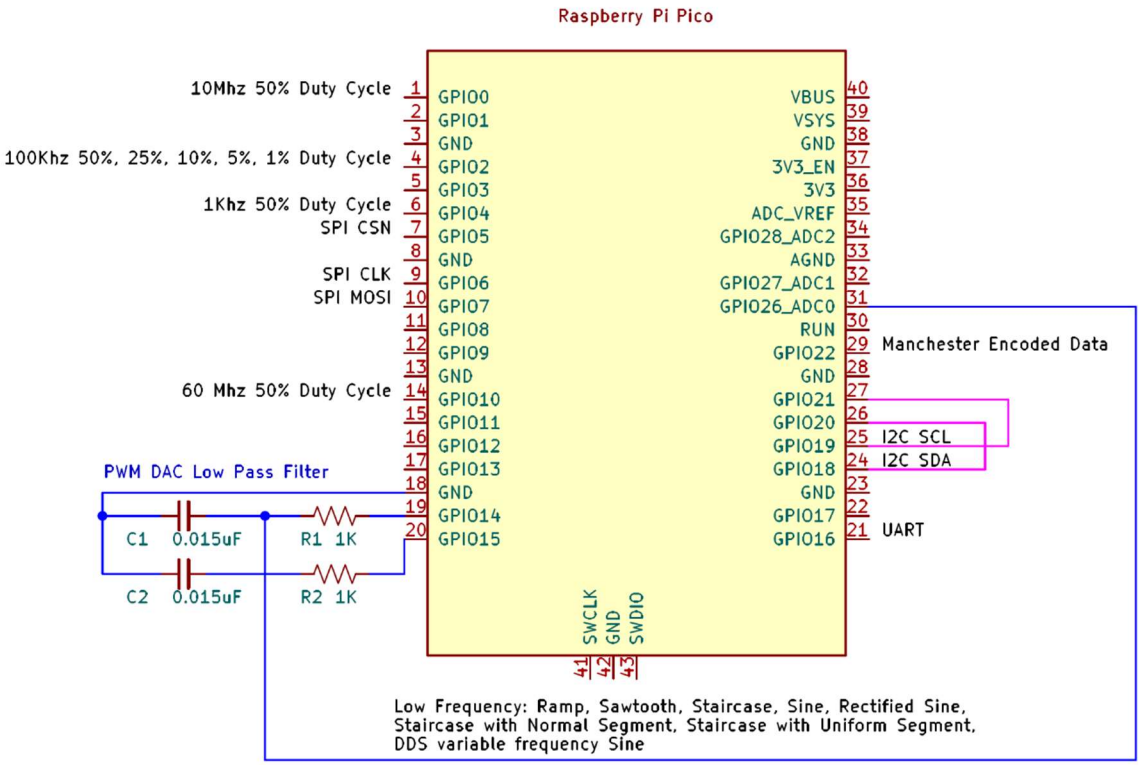
Introduction

I recently decided that I wanted a modern reasonably capable oscilloscope. I purchased one and sadly after an hour with it decided to return it (channel 3 was nonfunctional out of the box). The return went well and I purchase a different model from a different company. While waiting for the new scope to arrive I decided that I needed a way to quickly test basic functionality when the new scope arrived. I looked at several scope manufacturers' demo/test boards and while they were appealing, they were also reasonably pricey (hundreds of dollars). After careful consideration, I decided to use that money for other things and create my own basic test board. My goals were to create an inexpensive (couple of $), expandable, and customizable system that would be usable by a wide audience. This document describes what I have created. For those who want to jump right into it without going through this document, I have created a single page cheat sheet which is included in this distribution.

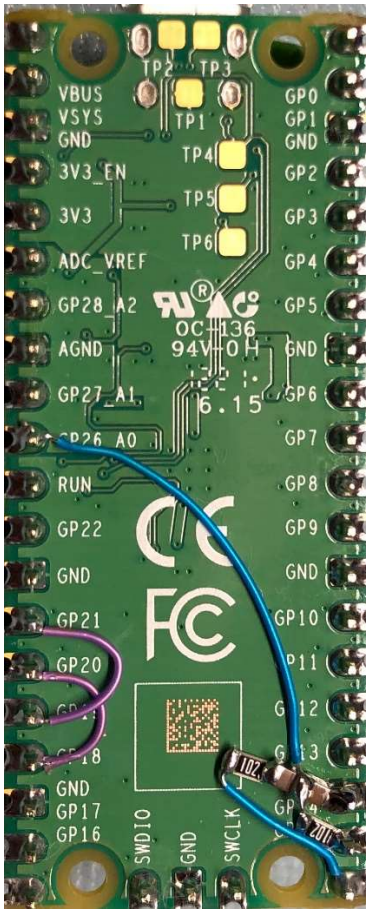
I would like to know if you find bugs, have suggestions for enhancements/changes, and/or if you found what I produced useful. You can reach me at pwproj24@gmail.com

Creation

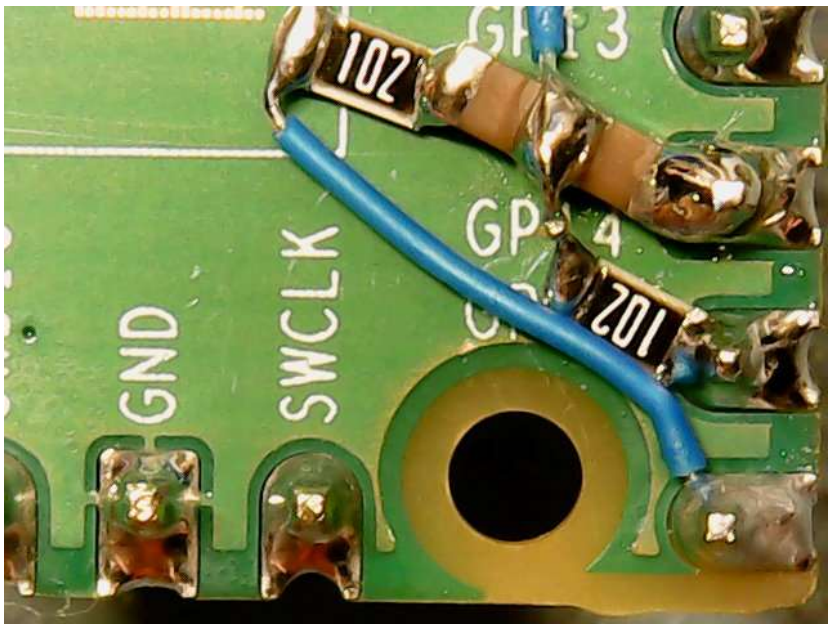
I decided to utilize a Raspberry Pi Pico. It can be obtained for $4 to $5. I would recommend "splurging" for the $5 version with the headers. You will need to make minor additions to the Pico for it to function. First, you will need to add some jumper wires along with two resistors and two capacitors. Below I have included a schematic illustrating the circuit and photos of my implementation.



Bottom View



Zoomed Bottom View



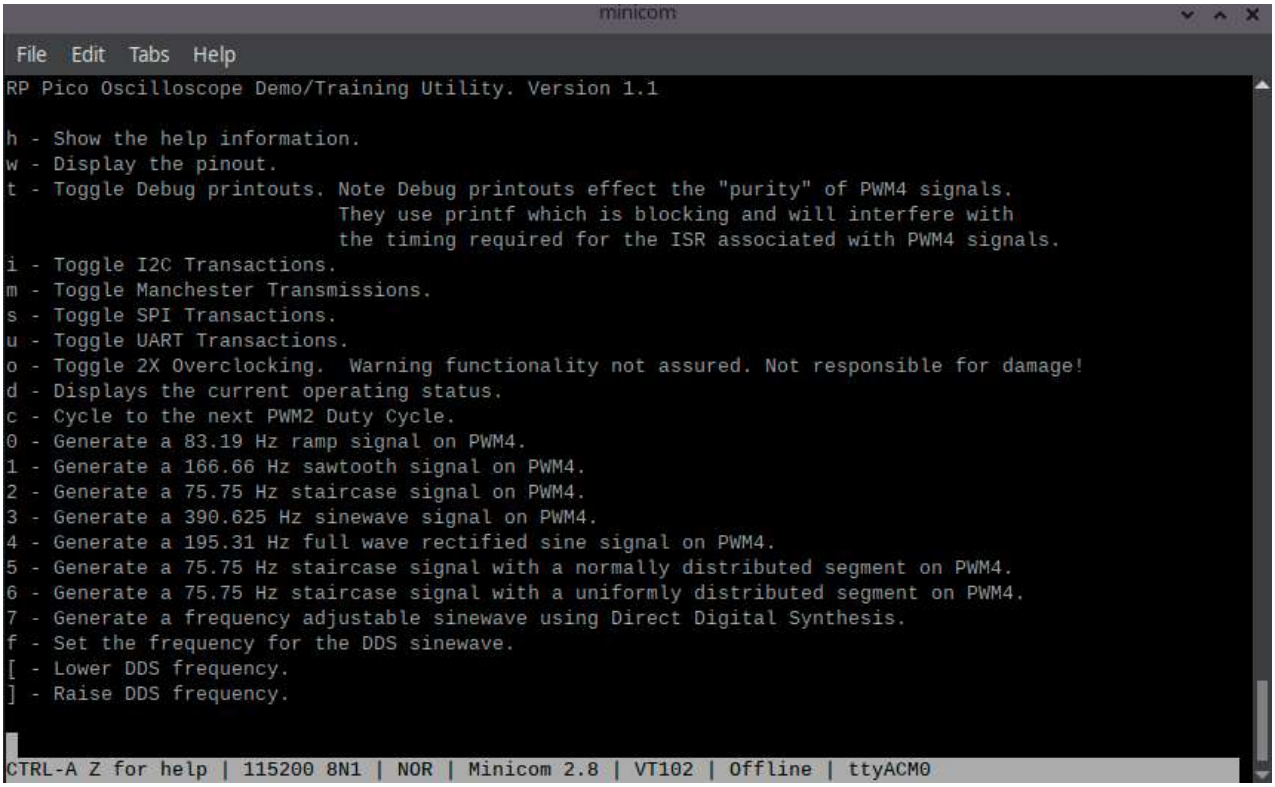
After the wiring is complete you will need to install the Sig_Gen program (sig_gen.uf2). If you are not familiar with this procedure, see: https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf for details on how to load this program. After this is completed, you are good to go. The uf2 file is included in this project's distribution.

Commands

The Sig_Gen program provides a command line interface. You should connect the Pico to a host computer (for power and serial over USB communications). I use minicom on my Raspberry Pi 5. I recommend Tera Term on a Windows PC. Again, if you are not familiar with Pico to host communications see the Getting Started Guide referenced above.

Commands are single characters (no Enter needed). The command list is shown in the screen shot below.

Note at start up: the I2C, Manchester, SPI, and UART signals are not enabled. You must use the commands listed below to enable them.



Functionality

Sig_Gen generates the following signals:

- 10/*20* MHz 50% Duty Cycle Waveform
- 100/*200* KHz 50%, 25%, 10%, 5%, 1% Duty Cycle Waveforms
- 1/*2* KHz 50% Duty Cycle Waveform
- 60/*120* MHz 50% Duty Cycle Waveform
- UART Waveform
- Manchester Encoded Data Waveform
- Low frequency PWM-DAC Analog Waveforms
    - 83.19/*166.38* Hz Ramp
    - 166.66/*333.32* Hz Sawtooth
    - 75.75/*151.5 Hz* Staircase
    - 390.62/*781.24* Hz Sine
    - 195.31/*390.62* Hz Full Wave Rectified Sine
    - 75.75/*151.5* Hz Staircase with a normally distributed center segment
    - 75.75/*151.5* Hz Staircase with a uniformly distributed center segment
    - 1.0-1000.0/*2.0-2000.0* Hz Frequency settable Sine
- I2C Waveforms
- SPI Waveforms

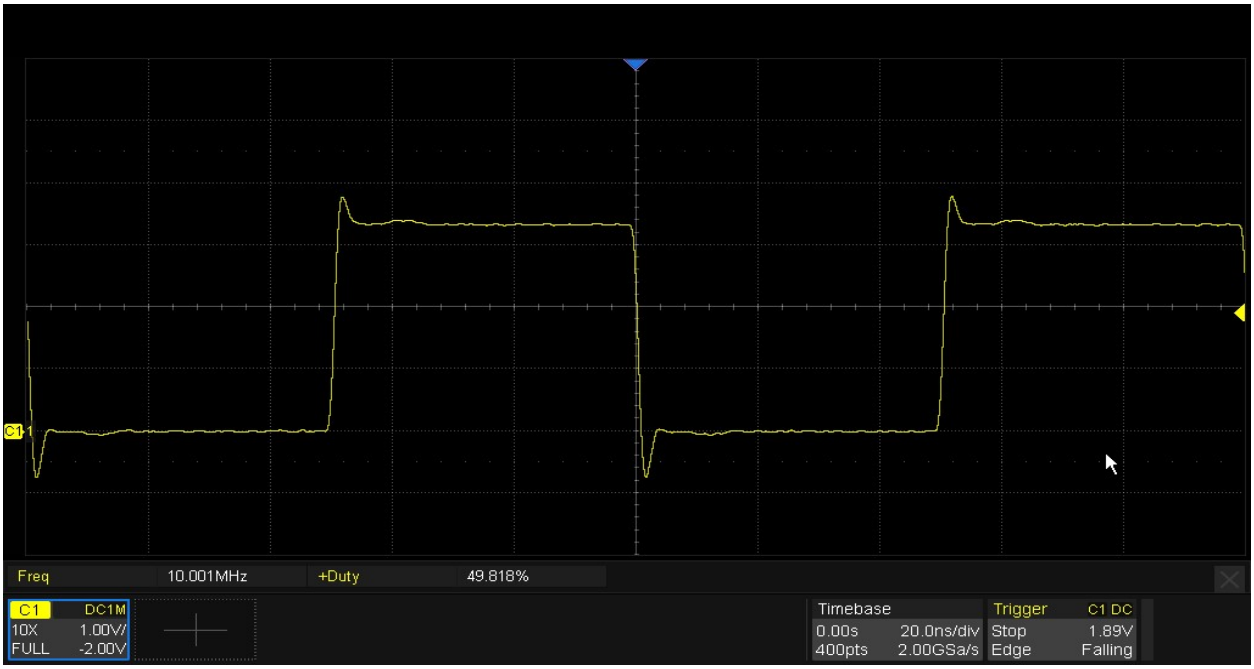
Note: The *Orange Italic* notation indicates the operating frequency when the 2X overclocking option has been enabled.

The following sections of this document provide additional details on each generated signal and shows scope traces of these signals.

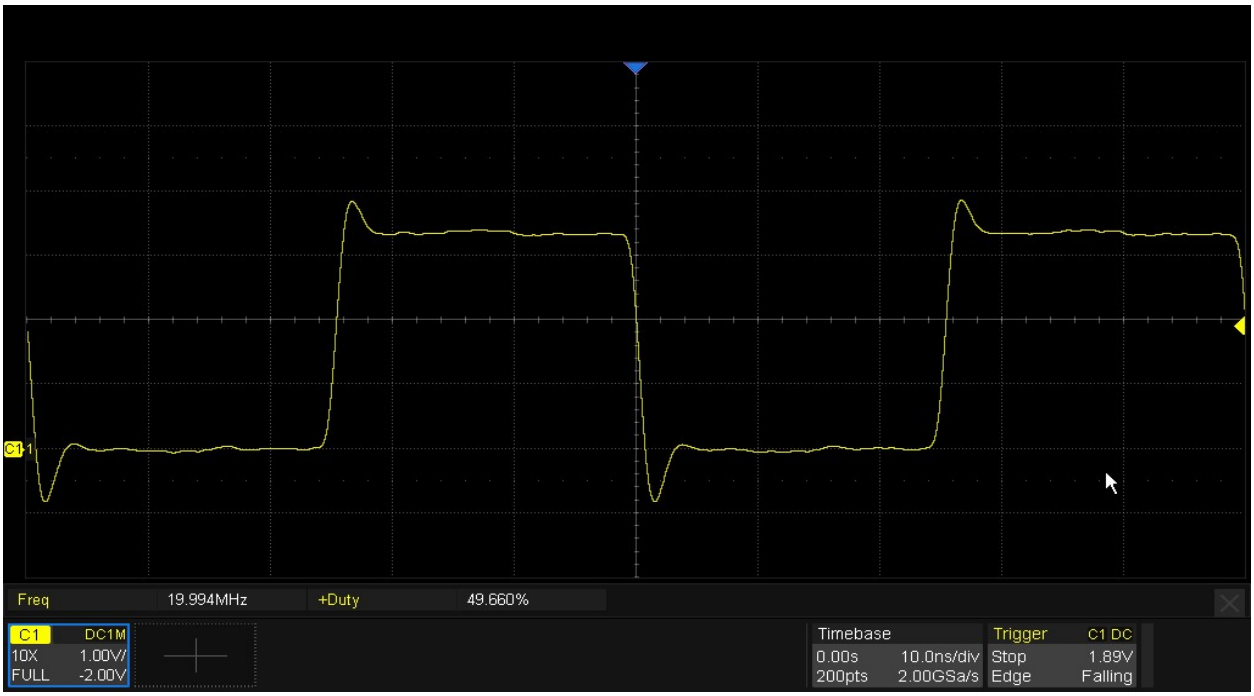
10/*20* MHz 50% Duty Cycle Waveform

The measurement techniques are important with "higher" speed signals. If you are new to using an oscilloscope, I recommend you review the Measurement Techniques section of this document.

10 MHz

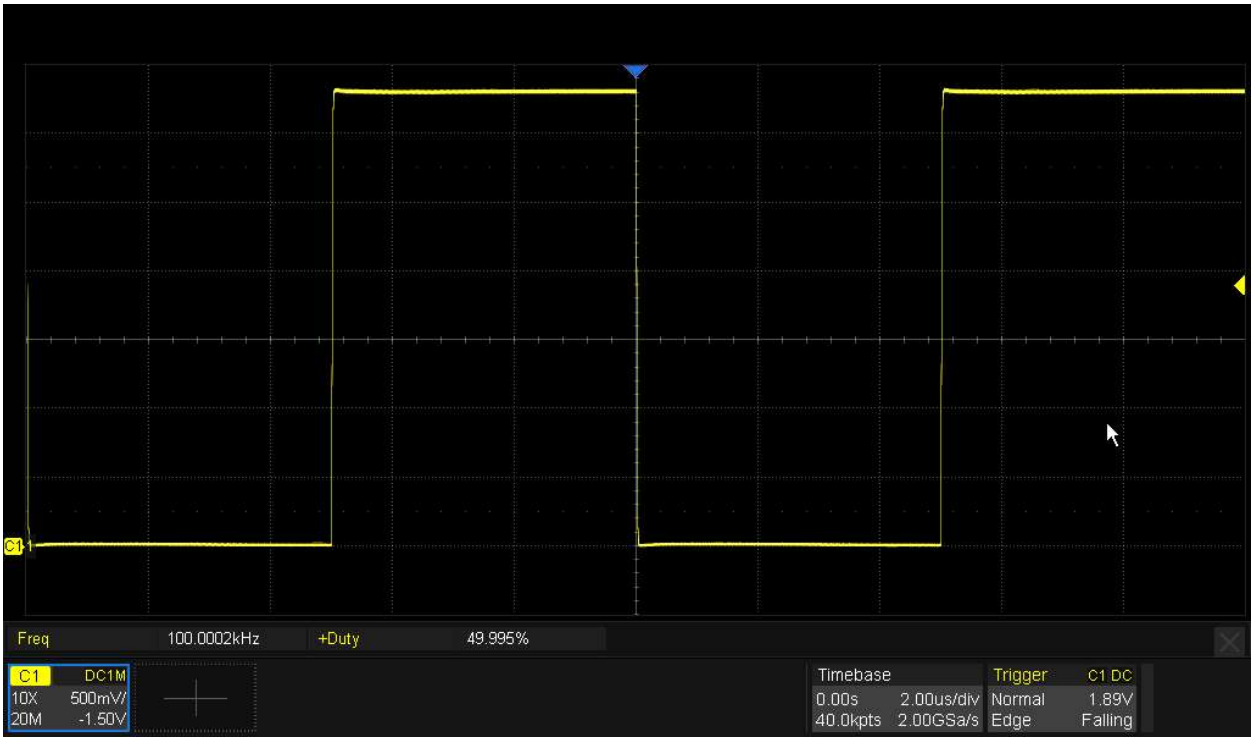


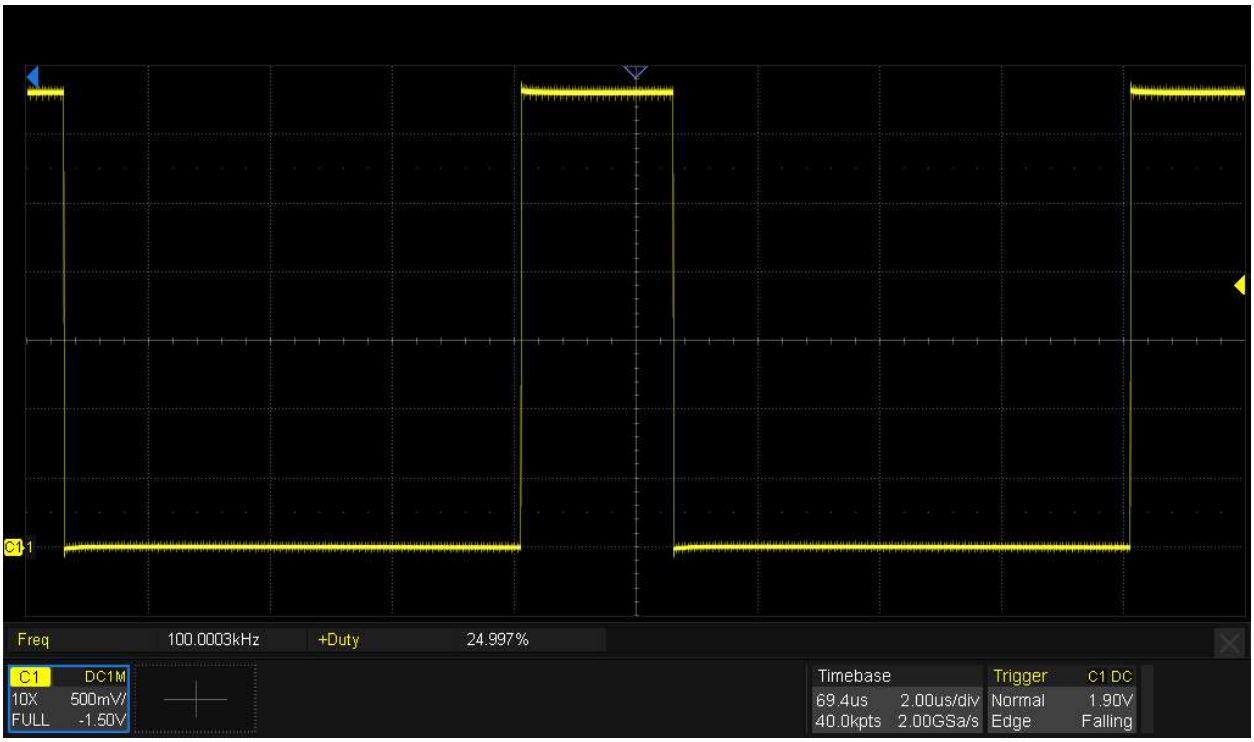
20 MHz

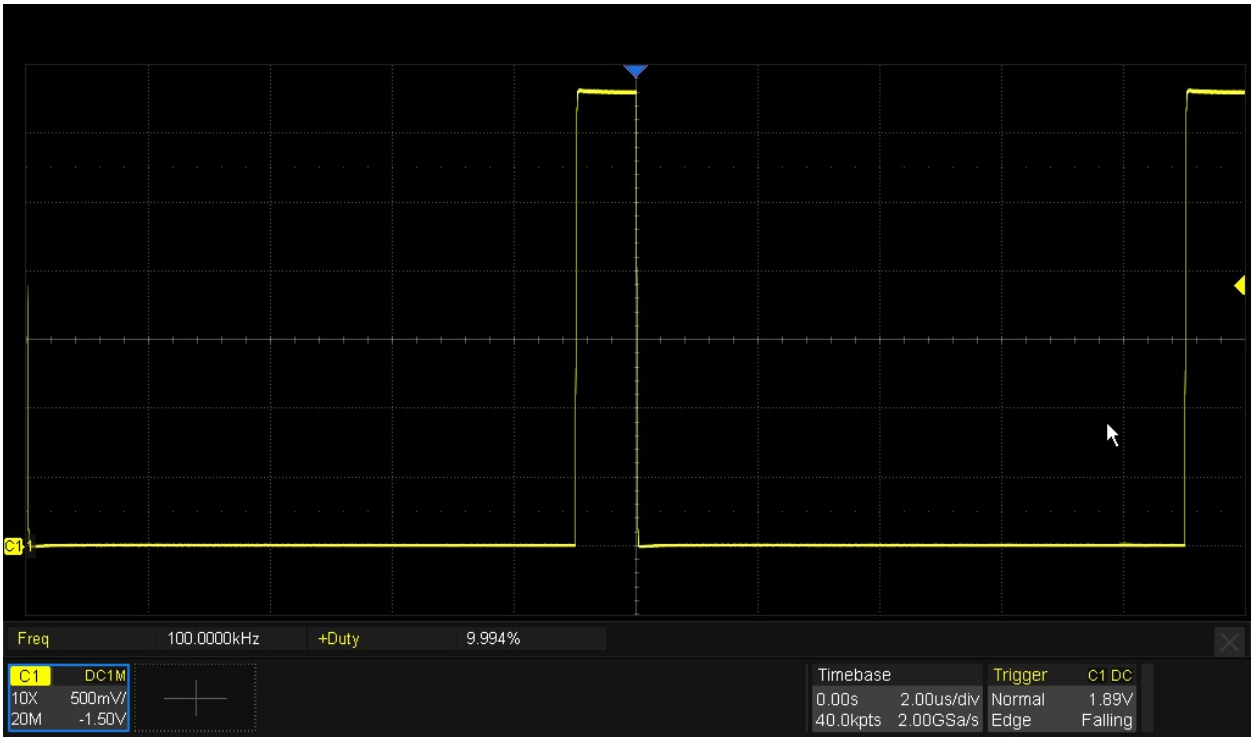


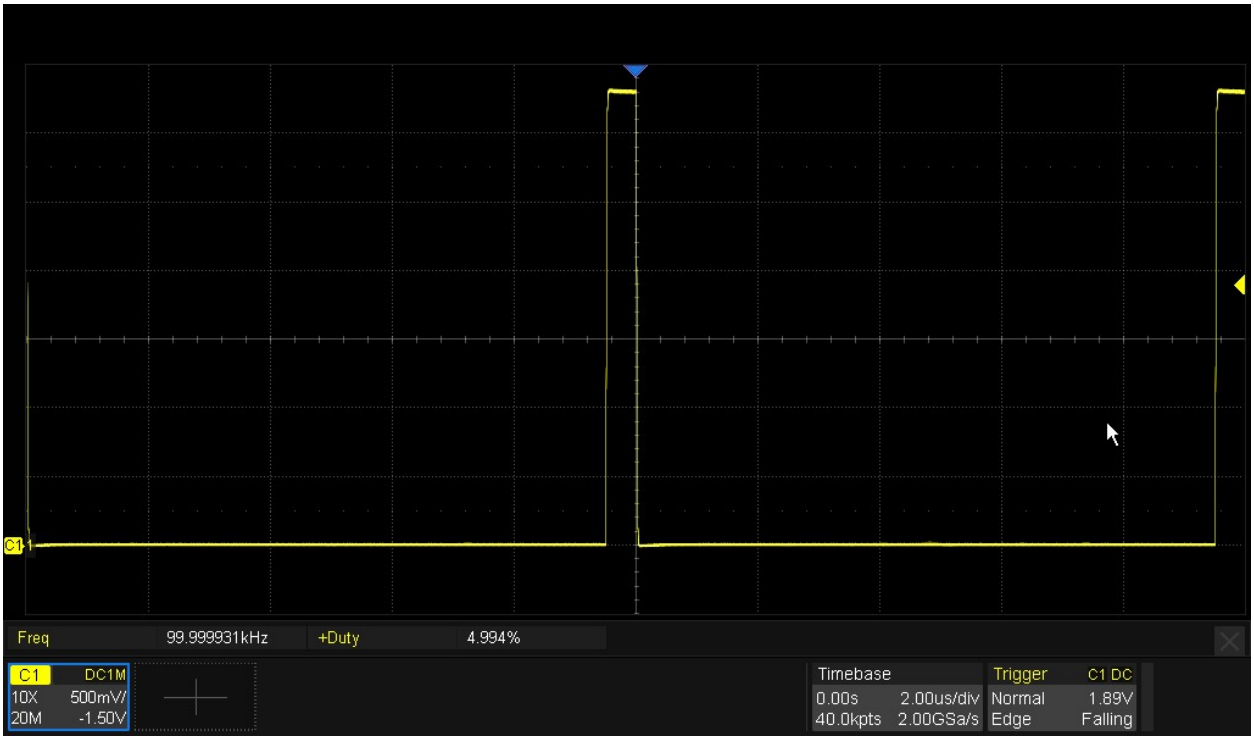
100/*200* KHz 50%, 25%, 10%, 5%, 1% Duty Cycle Waveforms

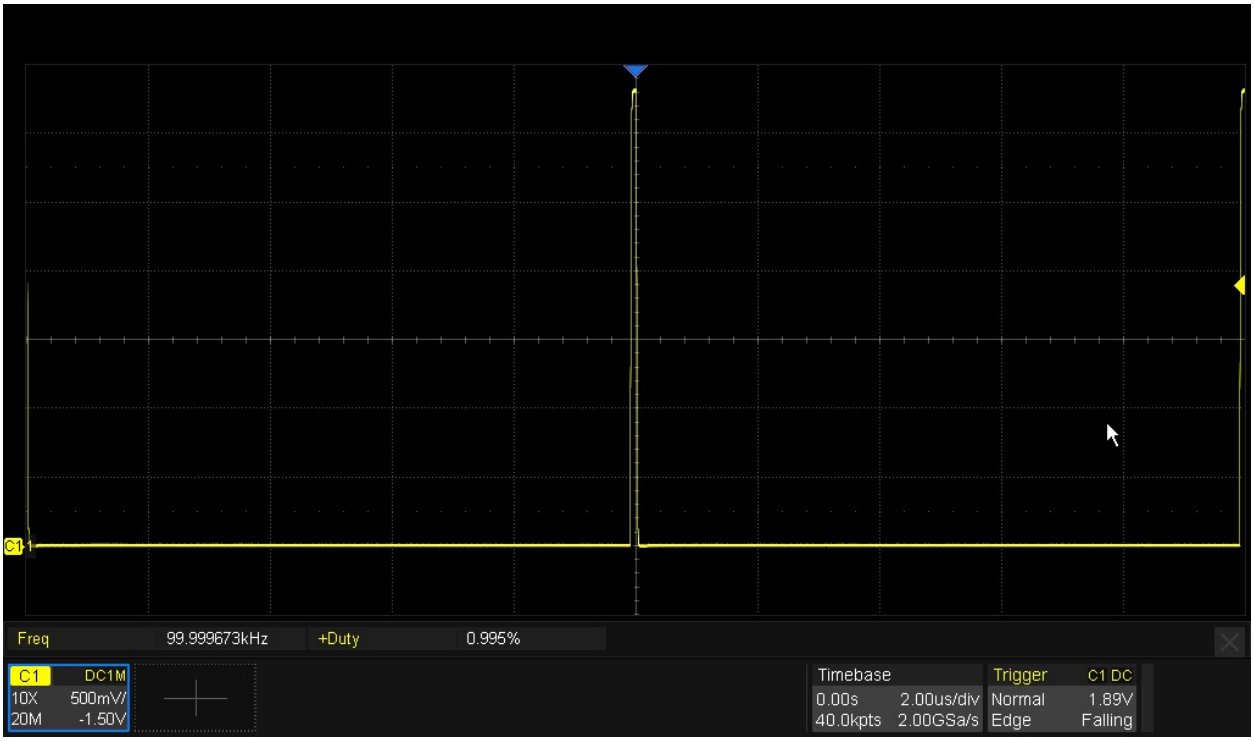
Use the “c” command to cycle through the various duty cycles. Only 100 KHz images are shown.





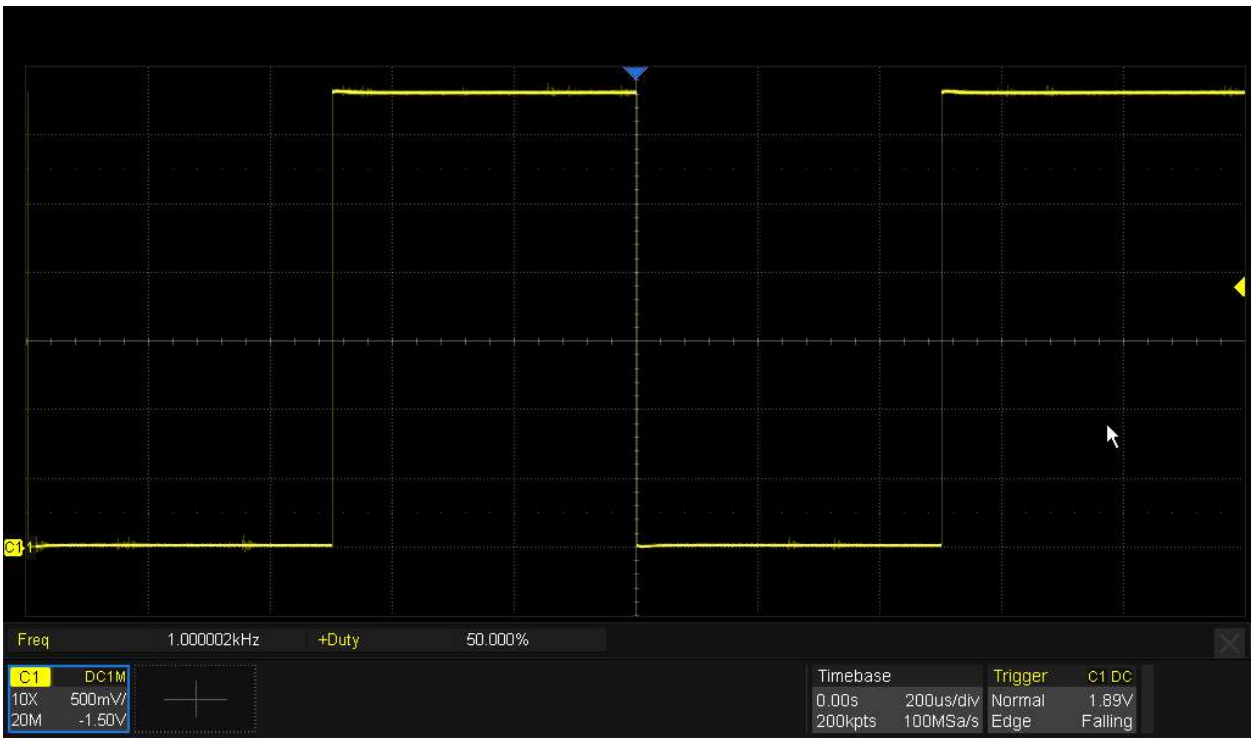

C1
Freq 100.0000kHz +Duty 9.994%
C1 DC1M
10X 500mV/
20M -1.50V
Timebase
0.00s 2.00us/div
40.0kpts 2.00GSa/s
Trigger C1 DC
Normal 1.89V
Edge Falling


C1
Freq 99.999931kHz +Duty 4.994%
C1 DC1M
10X 500mV/
20M -1.50V
Timebase
0.00s 2.00us/div
40.0kpts 2.00GSa/s
Trigger C1 DC
Normal 1.89V
Edge Falling



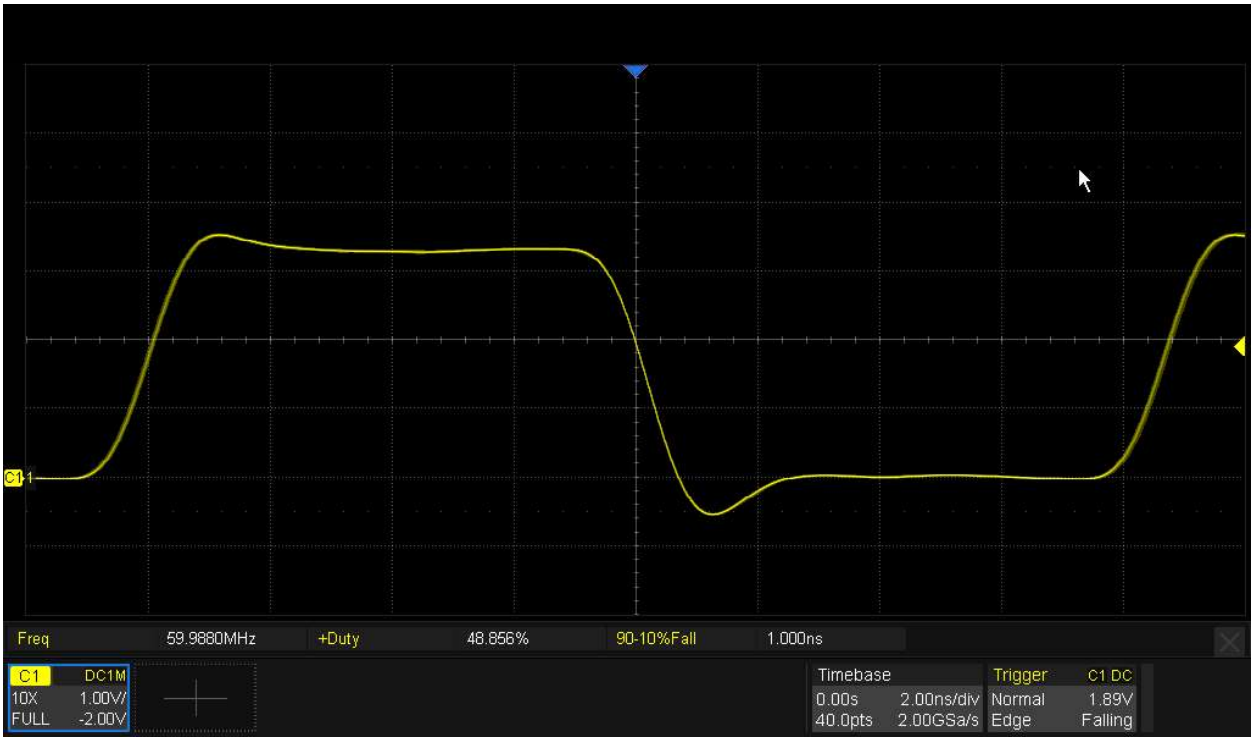
1/2 KHz 50% Duty Cycle Waveform

Only 1 KHz image shown.

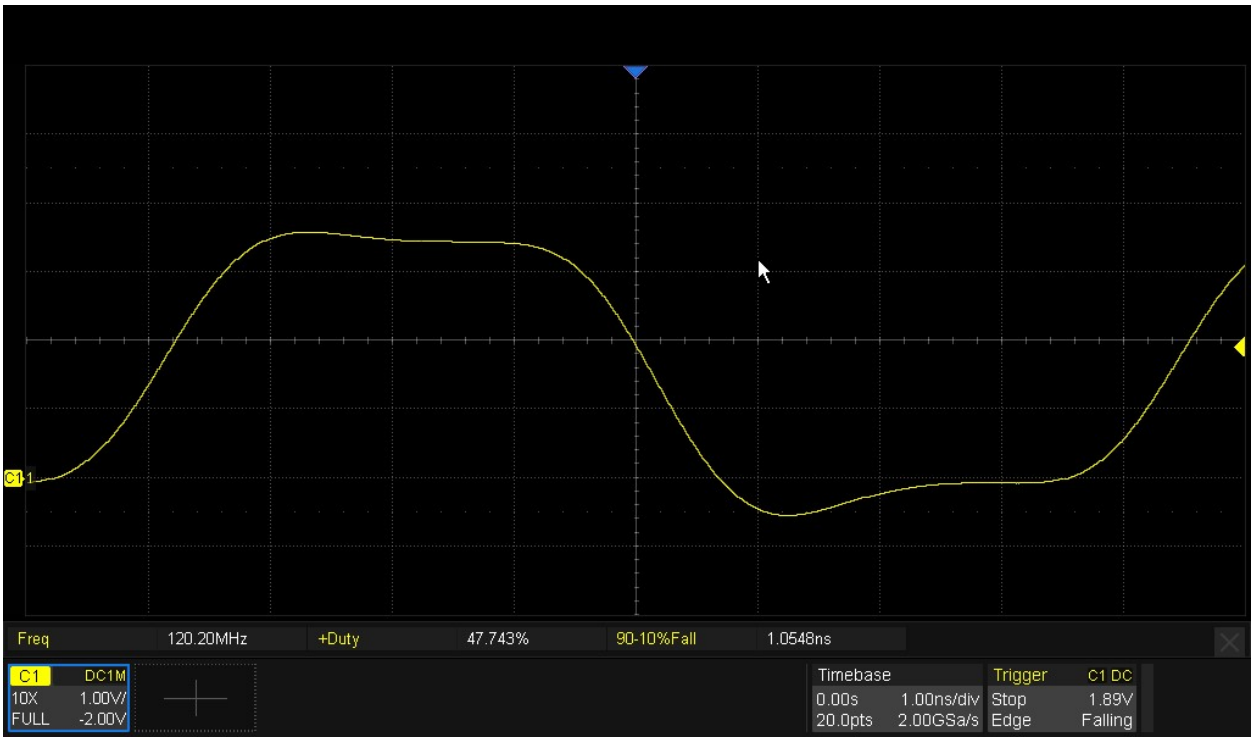


60/*120* MHz 50% Duty Cycle Waveform

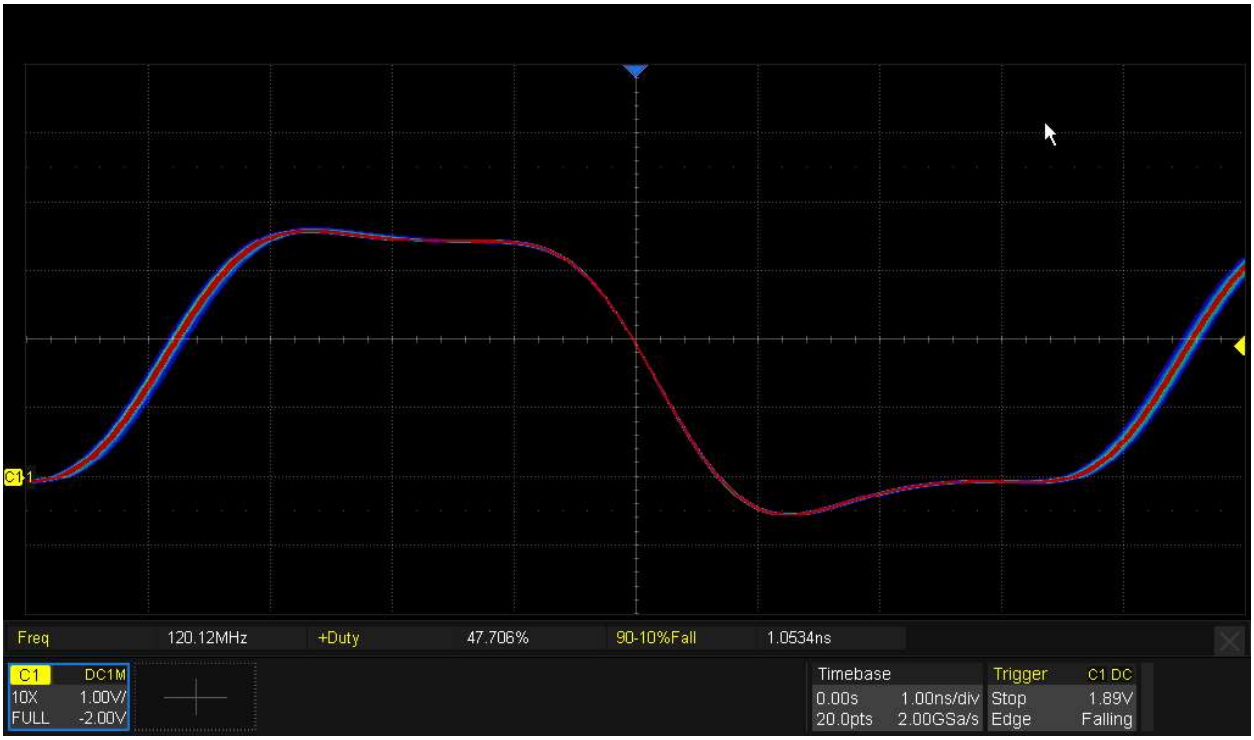
60 MHz



120 MHz



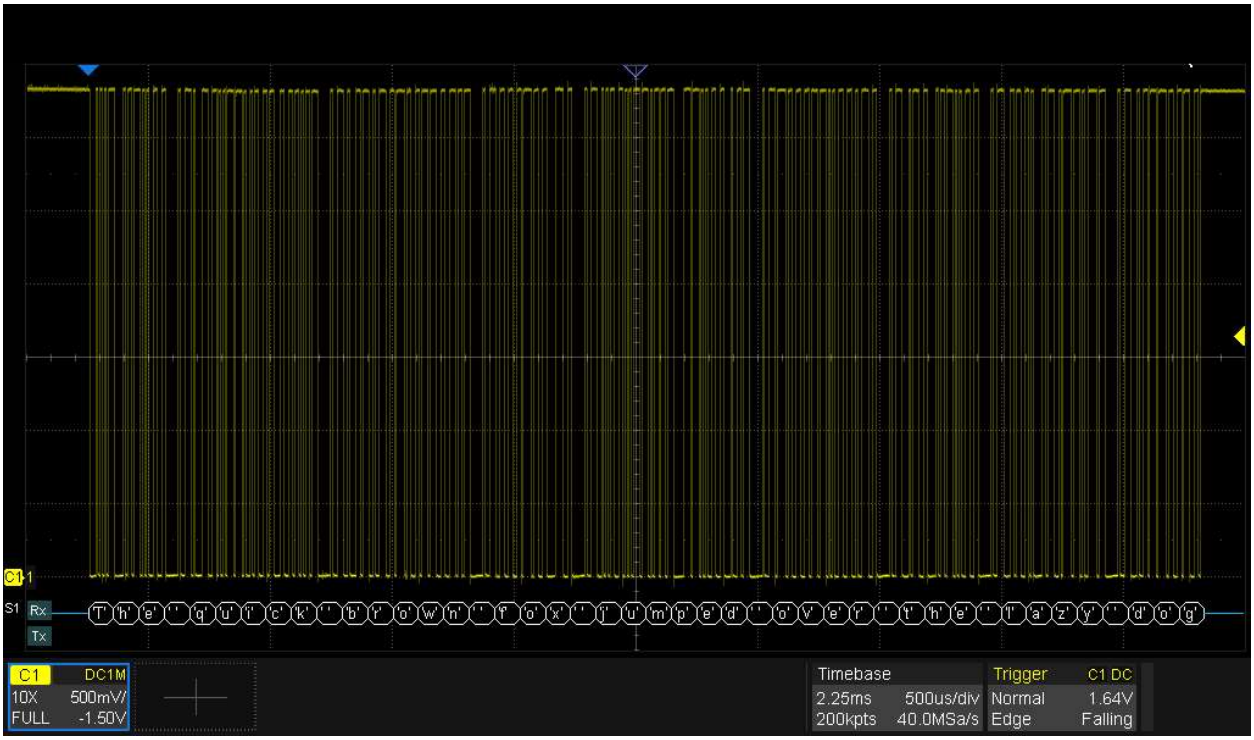
120 MHz with Color Grading enabled



UART Waveform

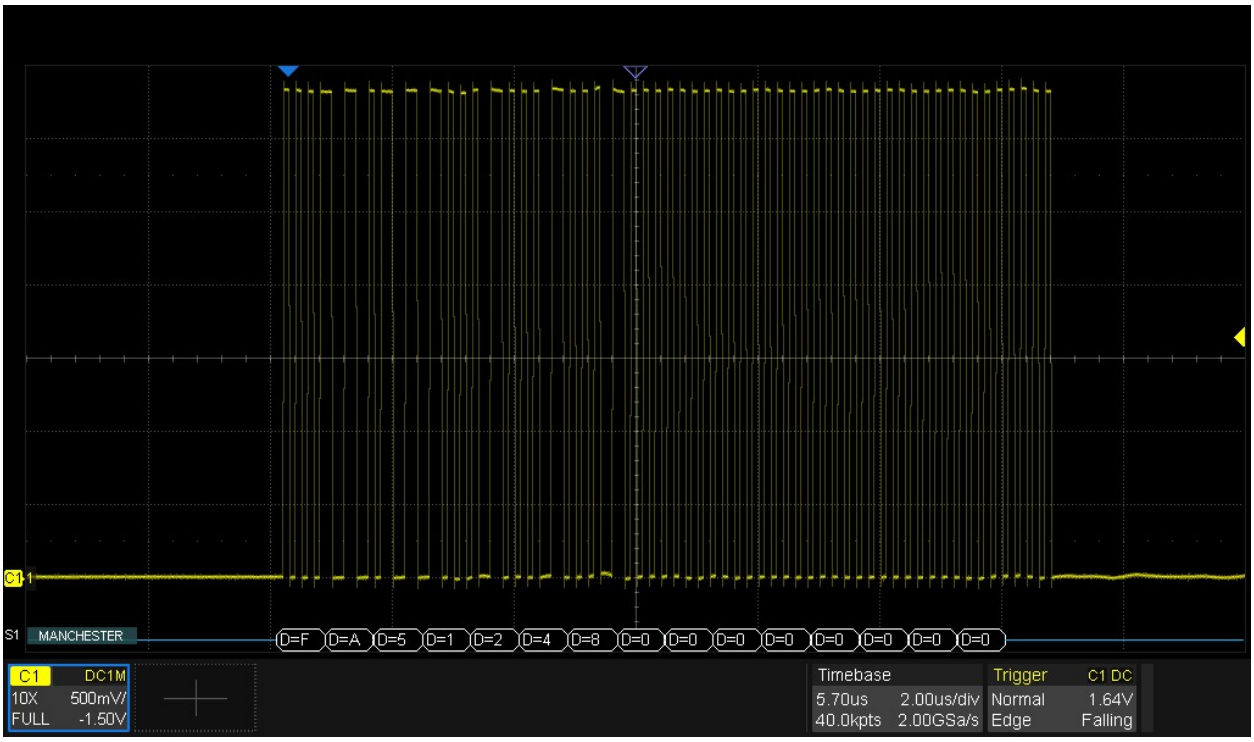
The UART is configured with 115.2 Kbps, 8 data bits, 2 stop bits, and even parity. No flow control.

The phrase “The quick brown fox jumped over the lazy dog” is transmitted.



Manchester Encoded Data Waveform

The following data is encoded 0x00000000084215AF and transmitted at 5/*10* Mbps. Note that this data is transmitted LSB first.

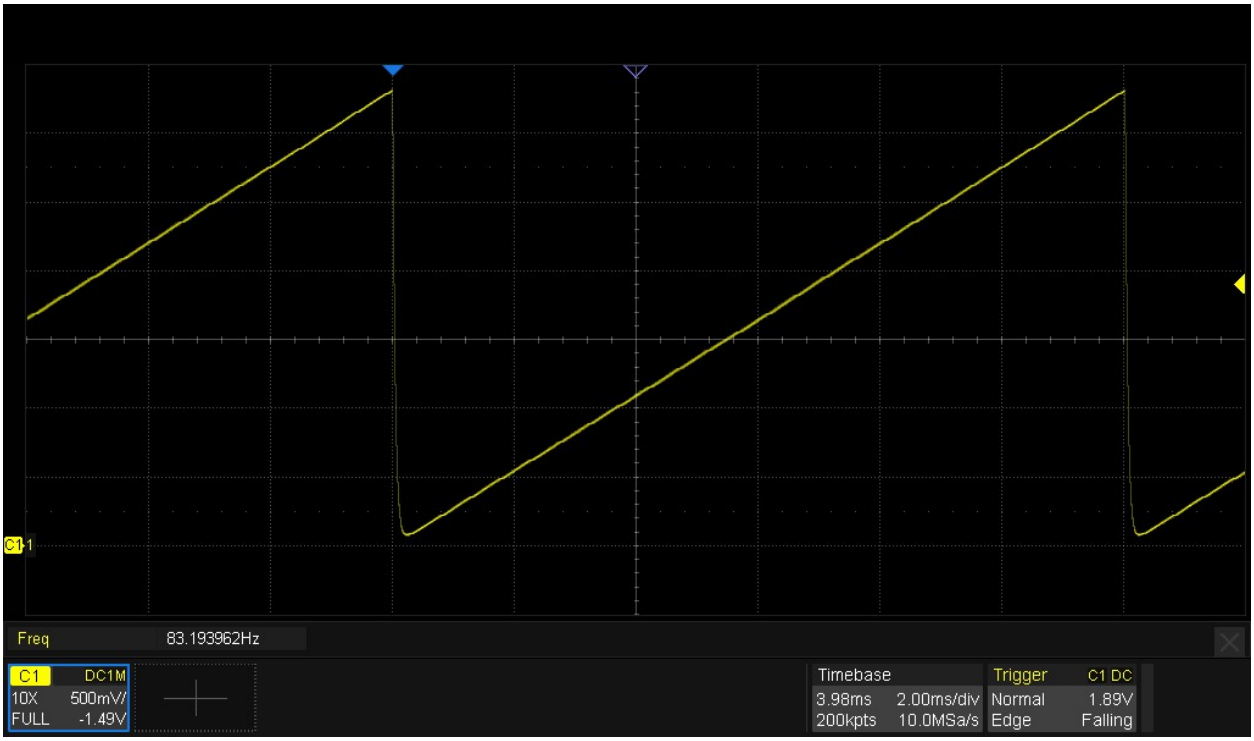


Low frequency PWM- DAC Waveforms

In version 1.1 of this program/hardware, a modulated PWM signal and a low pass RC filter are utilized to make a simple DAC. With this DAC, various "low" frequency signals are generated. Use commands 0-7 to cycle through the waveform types. The frequency of these signals will double when the 2X overclocking feature is enabled. This section of the document will only show the non-overclocked (1X) scope traces.

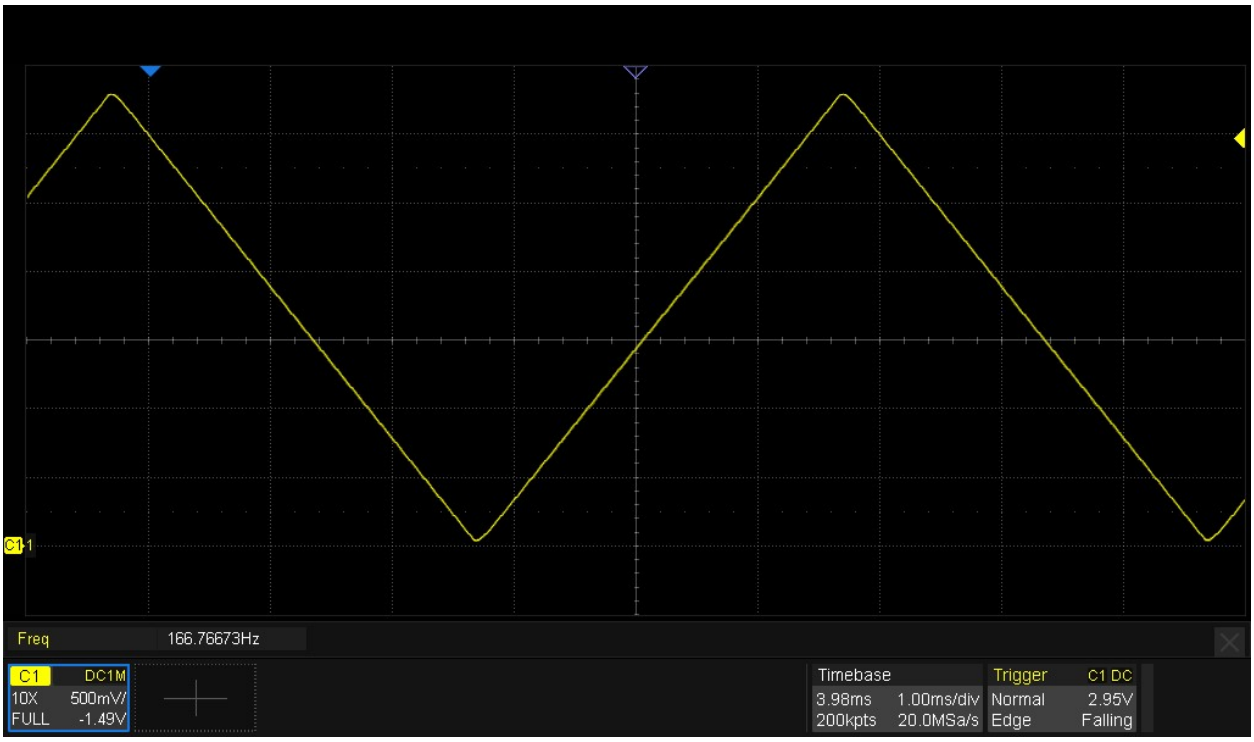
Ramp

Command "0" – 83.19/*166.38* Hz Ramp.



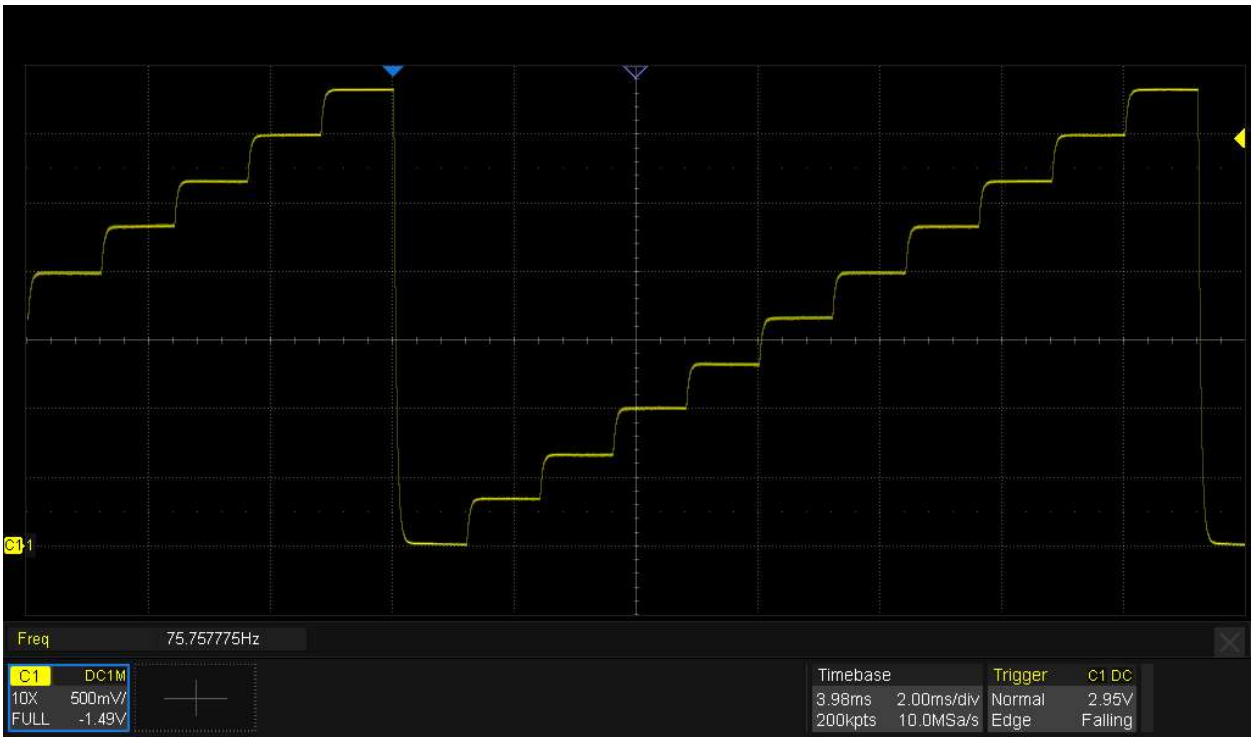
Sawtooth

Command “1” - 166.66/*333.32* Hz Sawtooth



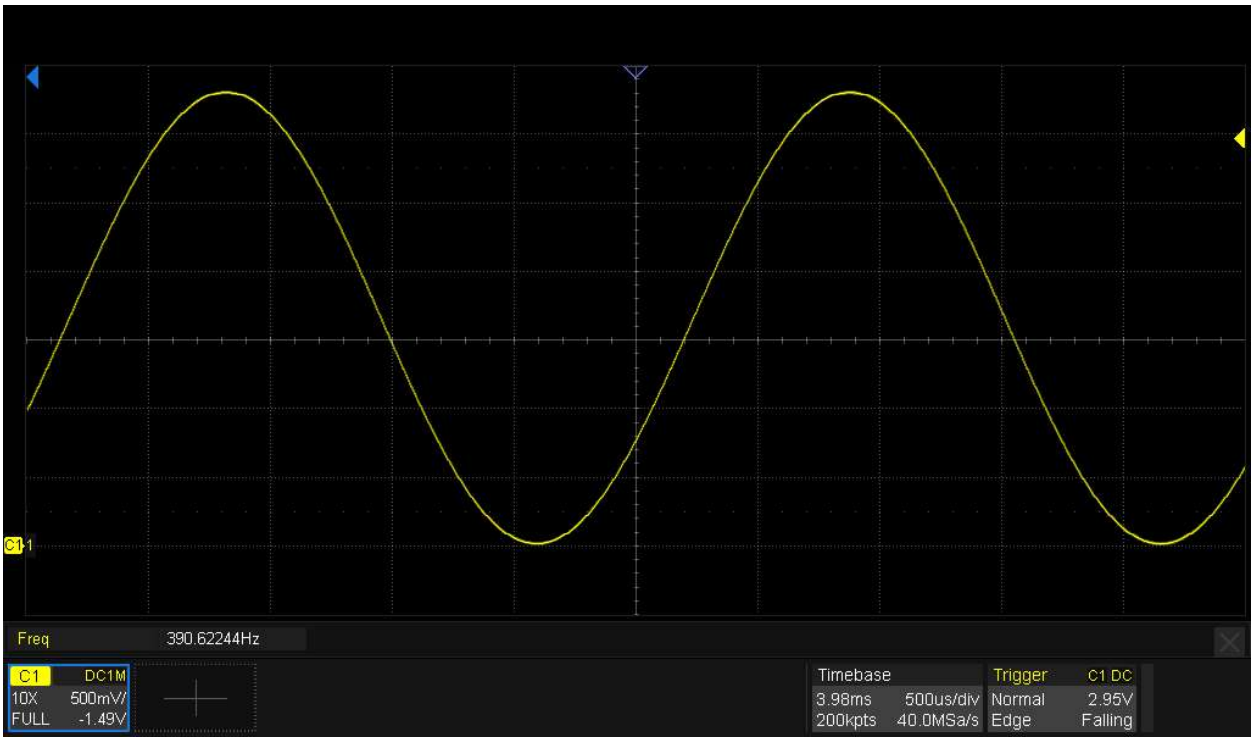
Staircase

Command “2” – 75.75/*151.5* Hz 11 Segment Staircase

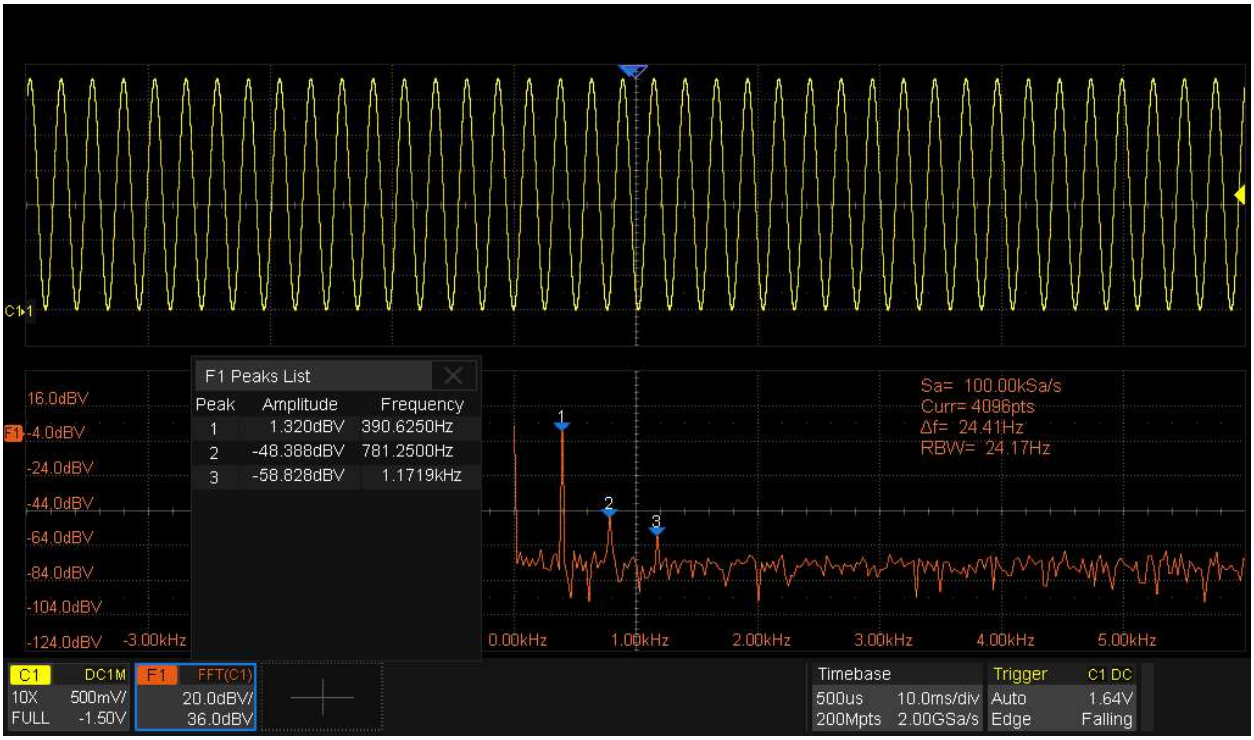


Sine

Command “3” – 390.62/*781.24* Hz Sine

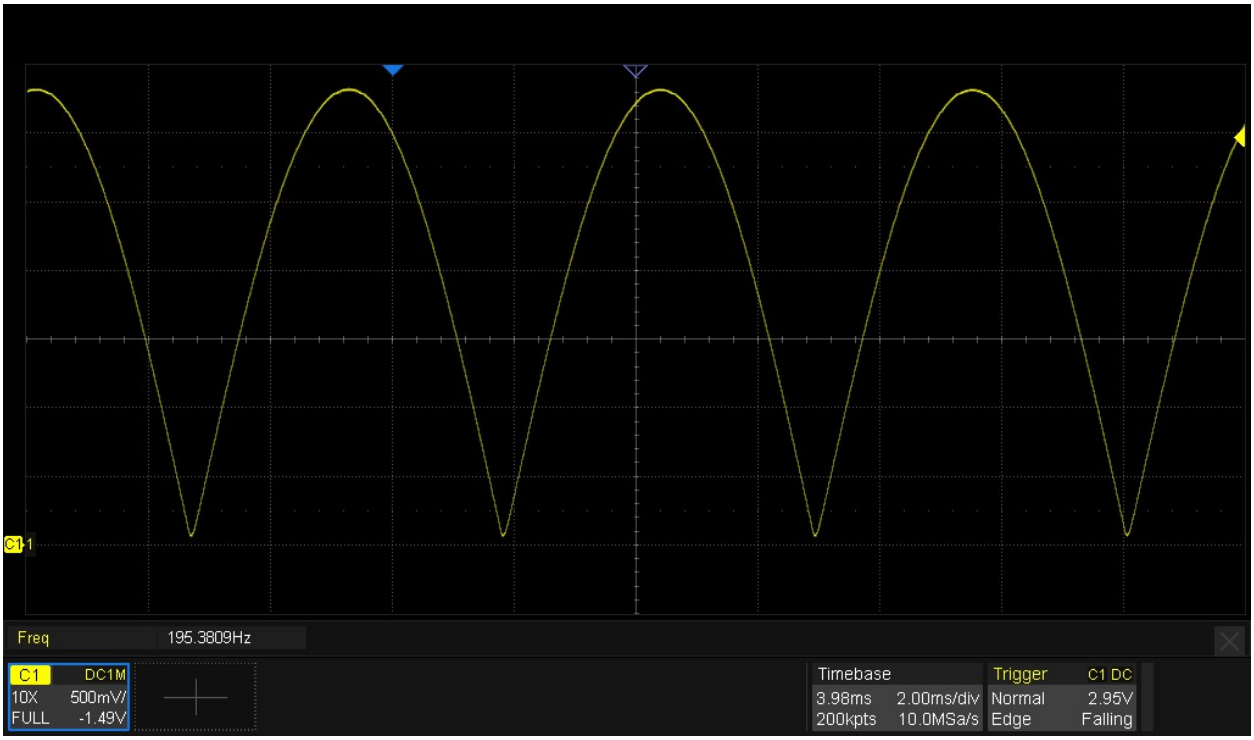


With FFT analysis enabled



Full Wave Rectified Sine

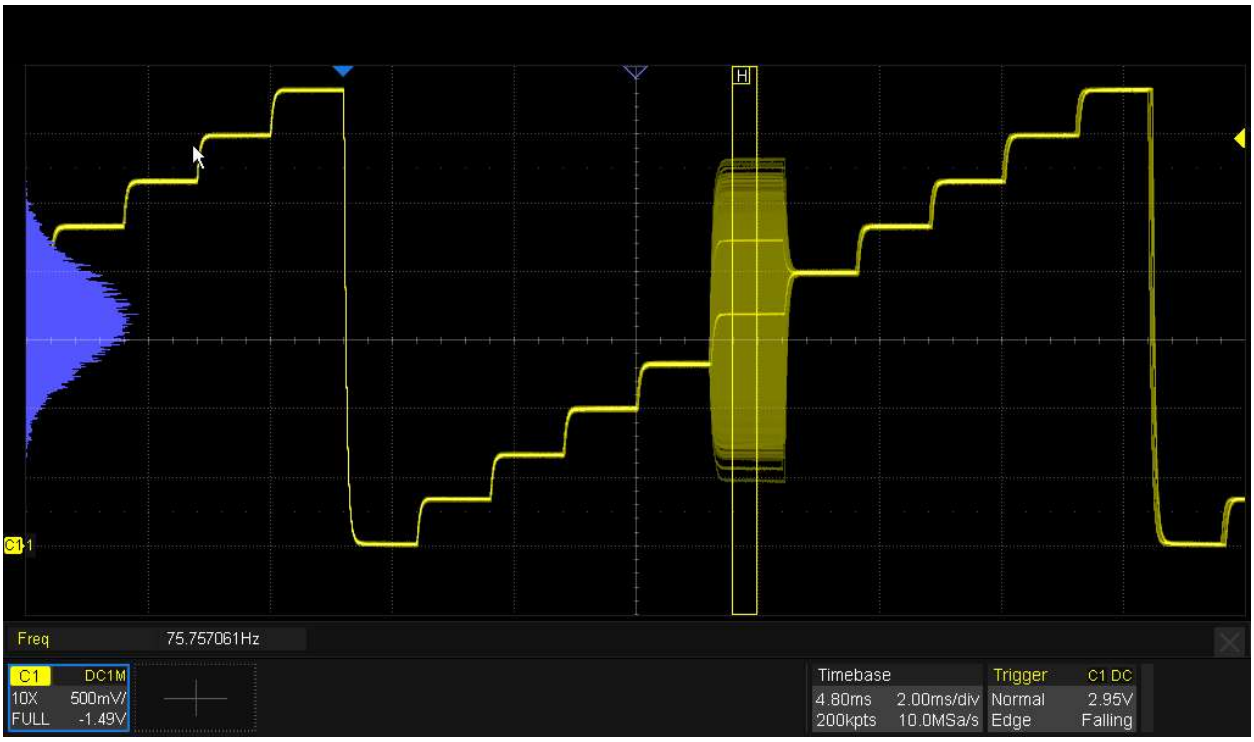
Command “4” - 195.31/*390.62* Hz Full Wave Rectified Sine



Gaussian Staircase

Command “5” - 75.75/*151.5* Hz 11 Segment Staircase with the middle segment level being Gaussian Distributed.

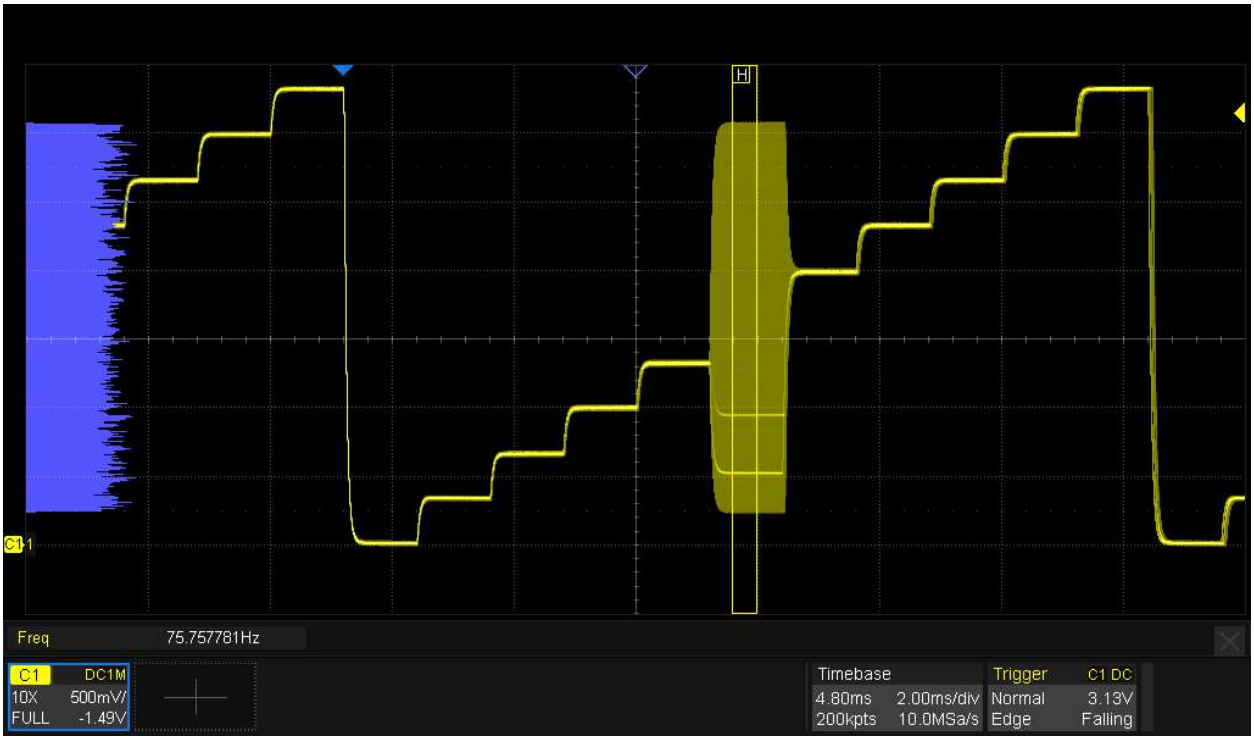
Note: the oscilloscope was set to infinite persistence and a histogram was enabled. The accumulation was run for some time to get a “reasonable” bell curve.



Uniform Staircase

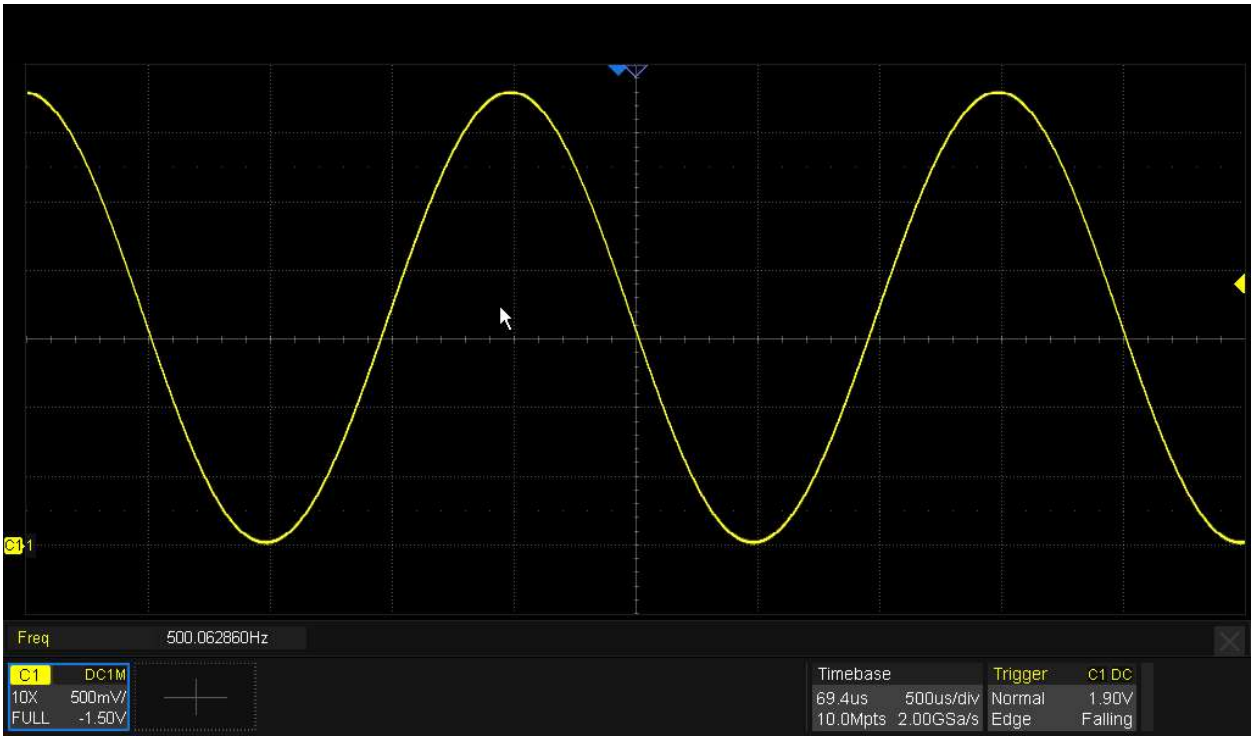
Command “6” - 75.75/*151.5* Hz 11 Segment Staircase with the middle segment level being Uniformly Distributed.

Note: the oscilloscope was set to infinite persistence and a histogram was enabled. The accumulation was run for some time to get a “reasonable” distribution.



Frequency Settable Sine

Command “7” - 1.0-1000.0/*2.0-2000.0* Hz



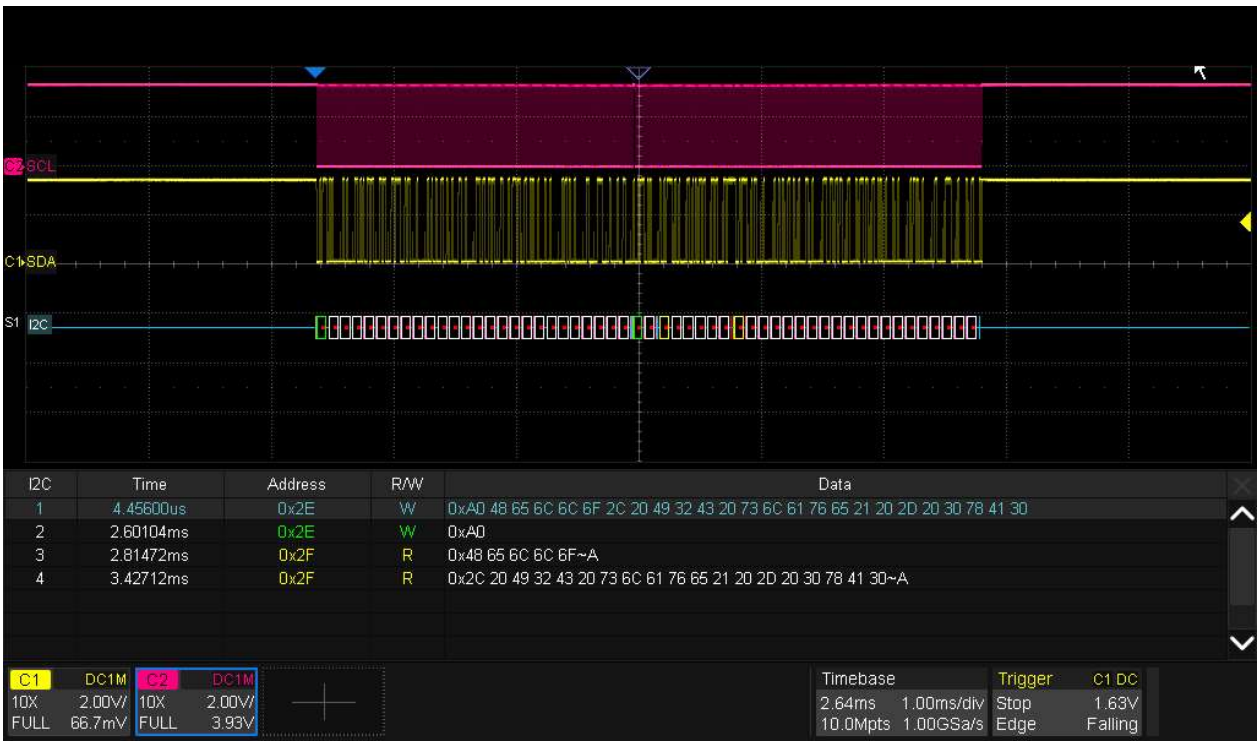
I2C Waveforms

100 Kbps, Slave Address 0x17 (0x2E W, 0x2F R)
Four transactions occur per loop.

Write	M + “Hello, I2C slave! - 0xm0”
Write	M
Read	“Hello”
Read	“, I2C slave! - 0xm0”

Where M is the binary value which cycles thru 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, 0xE0.
m is the ASCII character that represents the most significant nibble of M.

Full View



Zoomed View


C2 SCL
C1 SDA
[2.00V/,-66.7mV]
[2.00V/,-3.93V]
Zoom [20.0us/div, 92.0us]
C2 SCL
C1 SDA
S1 I2C ST W=0x2E A D=0x80
I2C Time Address R/W Data
1 4.45200us 0x2E W 0x80 48 65 6C 6C 6F 2C 20 49 32 43 20 73 6C 61 76 65 21 20 2D 20 30 78 38 30
2 2.60124ms 0x2E W 0x80
3 2.81496ms 0x2F R 0x48 65 6C 6C 6F~A
4 3.43028ms 0x2F R 0x2C 20 49 32 43 20 73 6C 61 76 65 21 20 2D 20 30 78 38 30~A
C1 DC1M 10X 2.00V/ FULL 66.7mV
C2 DC1M 10X 2.00V/ FULL 3.93V
Timebase 2.64ms 1.00ms/div 10.0Mpts 1.00GSa/s
Trigger C1 DC Stop 1.63V Edge Falling

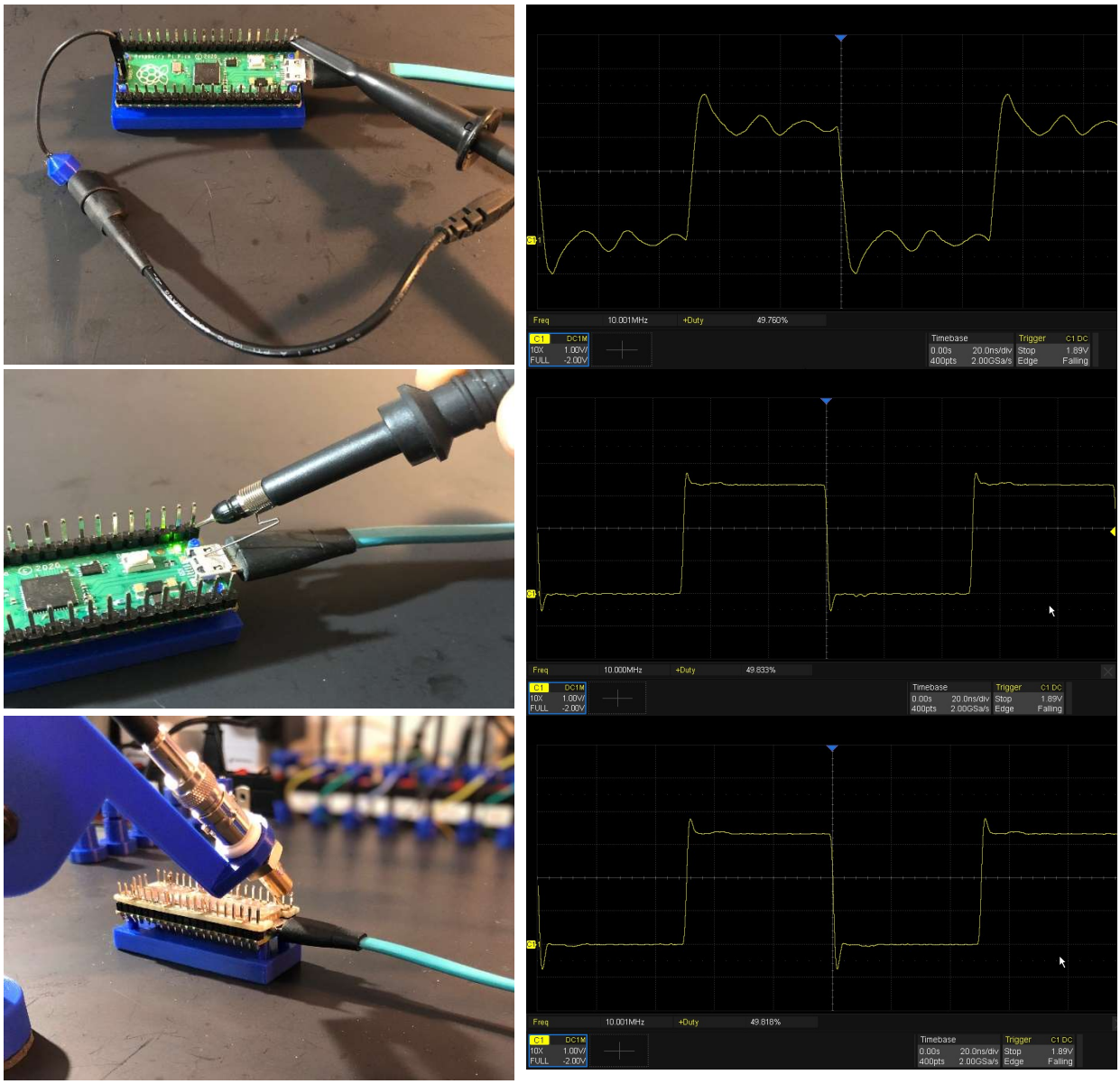
SPI Waveforms

1 Mbps

Transmitted message: ”RP”+ 16-bit incrementing loop counter + 12 bytes of sequential incrementing data. I.e. 16 total bytes per main loop.

Measurement Techniques

If you are a **new** scope user and are trying to match the pictures presented here, I have some advice for making measurements of the “high” speed signals. How you measure it matters. See the following set of pictures.

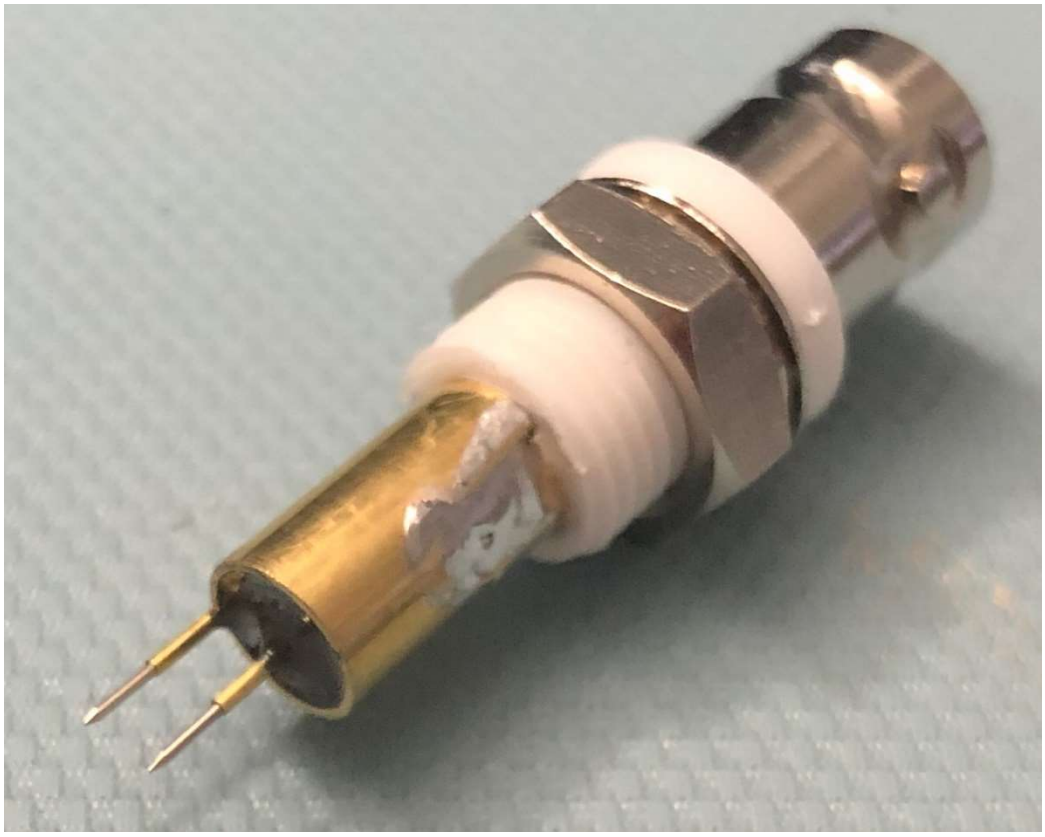


The first row uses the scope’s long ground lead (and an attachment wire). It is inductive and the resultant scope trace looks badly distorted. The second row utilizes the “classic” scope spring clip connected to a nearby grounded spot (the USB connector shell). This ground spring clip greatly reduces the path inductance and the resultant scope trace is much less distorted. The downside of this approach is that if you want to simultaneously measure multiple signals, it gets extremely difficult to hold all the probes. For the measurements made in this document, I created custom scope probe holders and low inductance BNC to sharp pogo pin adapters. This was used along with a low inductance ground plane temporarily added to the top of the Pico. The resultant signal (shown on the third row) closely matches the spring clip

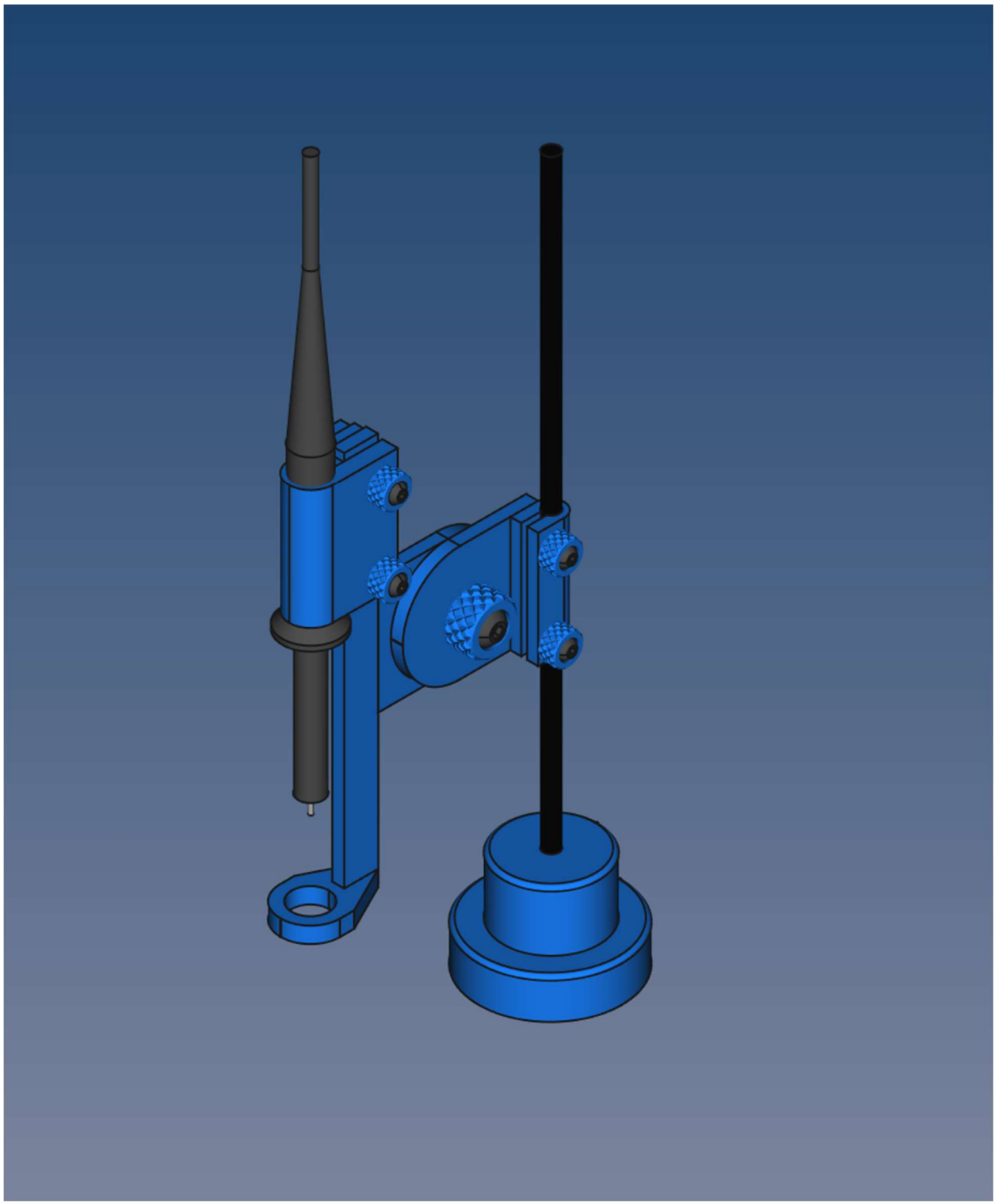
approach without the need to hold the scope probe. For the SPI measurements, three probes were needed and all were held with the custom designed setup.

Custom Measurement Jig Photos

Custom BNC chassis panel to spring pogo pins adapter

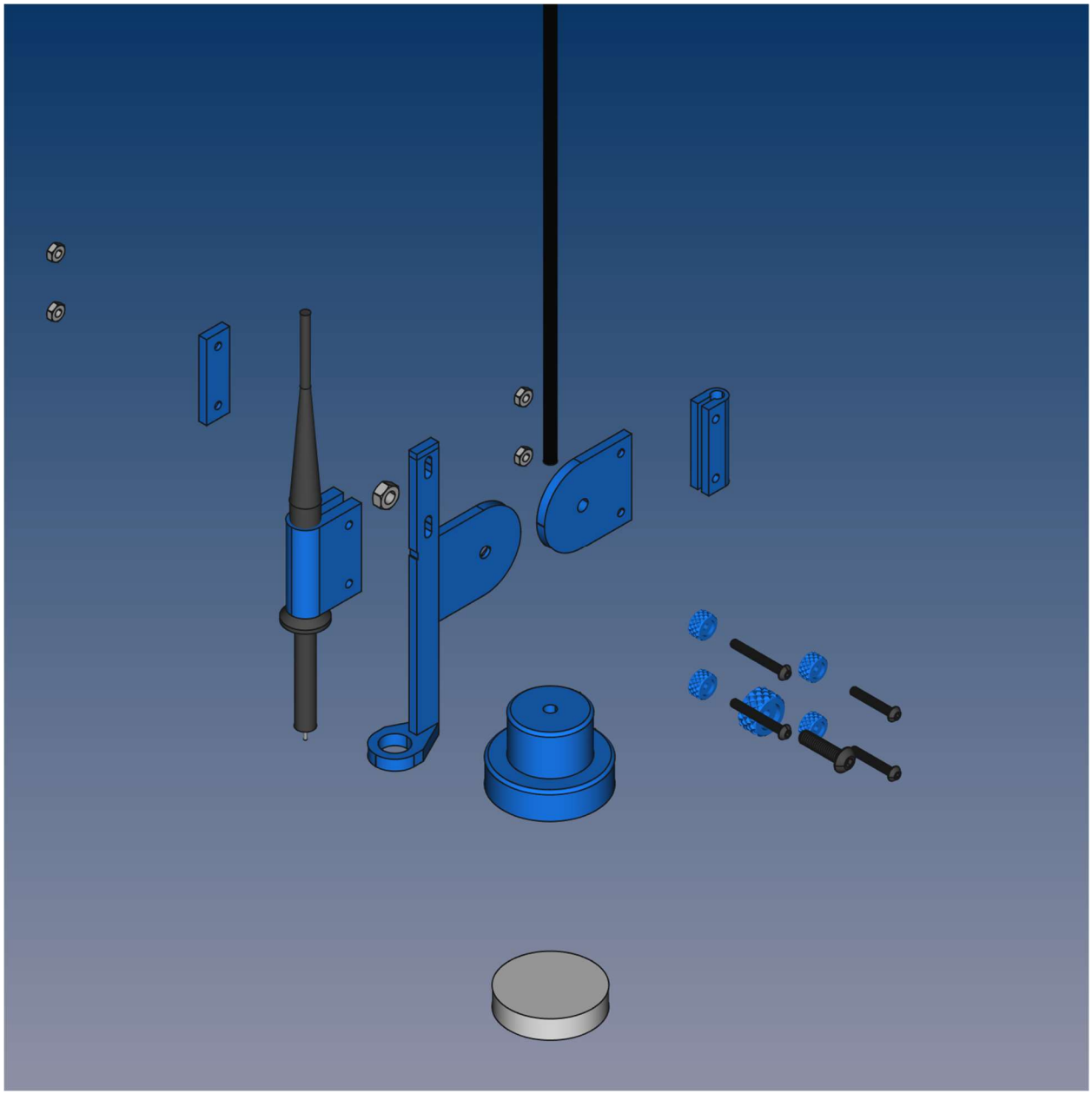


Design of custom scope probe holder



This is my design except for the knurled-knob which I reused. I found the knob on Thingiverse and it was created by user Perinski, and is licensed under Creative Commons – Attribution.

Exploded View



Scope probe holder in use