THAT Analog Computer Book

Collection of Examples, useful Hints and Electronic Circuits

by

Michael Koch, astroelectronic@t-online.de

Version from June 9, 2025

Note: Due to a bug in Libre Office / PDF export, the numbers in the table of contents may be wrong by a few pages.

Table of Contents

1 Analog Computers

1.1 Useful Links for THAT

THAT Website: https://the-analog-thing.org/

Wiki: https://the-analog-thing.org/wiki/Main_Page

Documentation: https://the-analog-thing.org/docs/dirhtml/

First steps book: https://the-analog-thing.org/THAT_First_Steps.pdf

List of application notes for analog computers: https://analogparadigm.com/documentation.html

Schematic diagrams of THAT: https://the-analog-thing.org/wiki/The_Analog_Thing#Schematics

Printed Circuit Board Files on Github: https://github.com/anabrid/the-analog-thing/tree/main/ANATHING_OSH

Anabrid Model 1 schematics on Github: https://github.com/anabrid/Model-1/tree/main/hardware

THAT Newsletter archive: https://the-analog-thing.org/newsletter/

Analog computer symbols for KiCad, by Arno Jacobs: https://github.com/Arno65/Analog-Computer-Symbols

Examples from Dietmar Hiller: https://pod.geraspora.de/tags/analogcomputer https://permondes.de/gitweb/Analog_Engine.git

Analog Computer Museum: https://analogmuseum.org/

Analog computer museum's Online Library: https://analogmuseum.org/english/library.html

Examples for analog computing: https://www.analogmuseum.org/english/examples/

Glen's Stuff: https://glensstuff.com/index.htm#5

1.2 Books for Analog Computing

Adler, Helmut: Elektronische Analogrechner (from 1962-1970)

Ammon, Werner: Schaltungen der Analogrechentechnik (from 1966, contains also circuits with resolvers)
https://www.analogmuseum.org/library/Ammon_Schaltungen.pdf

Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

EAI Handbook of Analog Computation (1971): https://analogmuseum.org/library/eai_handbook.pdf

This is an older version of the above book (1967): http://www.bitsavers.org/pdf/eai/EAI_Handbook_of_Analog_Computation_1967.pdf

Giloi, Wolfgang / Lauber, Rudolf: Analogrechnen (german) Pages 259 - 264: Simulation for a guitar string. The waveforms can be calculated simultaneously at different positions along the string.

Korn, Granino A. and Korn, Theresa M.: Electronic analog and hybrid computers ISBN 9780070353633
https://archive.org/details/electronicanalog0000korn_m6b3/page/n11/mode/2up

Leybold-Heraeus, Der Analogrechner (german): https://analogmuseum.org/library/der_analogrechner.pdf

Stice, James Edward: Electronic analog computer primer: https://archive.org/details/electronicanalog0000stic

Systron Donner, Handbook of Analog Computation: https://analogmuseum.org/library/handbook_of_analog_computation.pdf

Tomovic, Rajko and Karplus, Walter J.: High-Speed Analog Computers The original version of this book was written in 1962, and "high speed" probably means "faster than mechanical servo multipliers".

Ulmann, Bernd: Analog and Hybrid Computer Programming https://www.amazon.de/Analog-Computer.../dp/3110662078

Ulmann, Bernd: Analog Computing https://www.amazon.de/Analog-Computing-Gruyter-Textbook-Ulmann/dp/311078761X

1.3 Other useful Books

Analog Devices: Nonlinear Circuits Handbook, Second Edition January 1976, ISBN: 0-916550-01-X

Analog Devices: The Best of Analog Dialogue, ISBN 0-9-916550-10-9

Analog Devices: High Speed Design Techniques, ISBN 0-916550-17-6

Analog Devices: OPAMP Applications Handbook https://www.analog.com/en/resources/technical-books/op-amp-applications-handbook.html

Analog Devices: Multiplier Application Guide (1978) https://www.analog.com/media/en/training-seminars/design-handbooks/ADI_Multiplier_Applications_Guide.pdf

Fiore, James M.: Operational Amplifiers and Linear Integrated Circuits - Theory and Application https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Operational_Amplifiers_and_Linear_Integrated_Circuits_-_Theory_and_Application_(Fiore)

Horowitz, Paul and Hill, Winfield: The Art of Electronics, Second Edition, ISBN 0-521-37095-7

Horowitz, Paul and Hill, Winfield: The Art of Electronics, Third Edition, ISBN 978-0-521-80926-9

Horowitz, Paul and Hill, Winfield: The Art of Electronics - The x Chapters, ISBN 978-1-108-49994-1

Laning, J. Halcombe and Battin, Richard H.: Random Processes in Automatic Control

Pease, Robert A.: Troubleshooting Analog Circuits, ISBN 0-7506-9499-8

Tietze, U. and Schenk, Ch.: Halbleiter-Schaltungstechnik, 9. Auflage, ISBN 3-540-19475-4

Tietze, U. and Schenk, Ch.: Halbleiter-Schaltungstechnik, 11. Auflage, ISBN 3-540-64192-0

Williams, Arthur and Taylor, Fred: Electronic Filter Design Handbook, ISBN 0-07-147171-5

The latest version of this book: http://www.astro-electronic.de/THAT_Analog_Computer_Book.pdf

My FFmpeg Book: http://www.astro-electronic.de/FFmpeg_Book.pdf

My Teensy Book: http://www.astro-electronic.de/Teensy_Book.pdf

My C# Programming Book (in german language): http://www.astro-electronic.de/CSharp_Buch.pdf

My book about time travel and science fiction (in german language): http://www.astro-electronic.de/Zeitreisen.pdf

1.4 Analog Devices "Analog Dialogue"

All issues from Analog Devices "Analog Dialogue": https://www.analog.com/en/analog-dialogue/archives.html

Volume 1, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-1/number-1/articles/volume1-number1.pdf

Page 6: Operational integrators

Page 9: Comparison of capacitor specifications

Volume 1, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-1/number-2/articles/volume1-number2.pdf

Page 14: Overload clamp circuit

Volume 2, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-2/number-1/articles/volume2-number1.pdf

Page 12: Ideal rectifier

Volume 5, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-5/number-2/articles/volume5-number2.pdf

Page 15: Amplitudes without delay, measuring sine wave amplitudes without filtering

Volume 5, Number 5: https://www.analog.com/media/en/analog-dialogue/volume-5/number-5/articles/volume5-number5.pdf

Page 14: Sine oscillator

Volume 8, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-8/number-1/articles/volume8-number1.pdf

Page 13: RMS Measurement, Crest Factor

Volume 8, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-8/number-2/articles/volume8-number2.pdf

Page 23: Errata for Nonlinear Circuits Handbook

Volume 11, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-11/number-1/articles/volume11-number1.pdf

Page 6: AD534 Multiplier

Page 18: Errata for Nonlinear Circuits Handbook

Volume 12, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-12/number-3/articles/volume12-number3.pdf

Page 16: Vector Generators DTM1716 and DTM1717

Volume 13, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-13/number-2/articles/volume13-number2.pdf

Page 4: A/D converter with seriel (cascade) Gray code

Page 11: High resolution differential temperature measurement with AD590

Page 18: Four quadrant divider

Volume 14, Number 2:

Page 18: Some notes about the high resolution differential temperature measurement from volume 13, number 2.

Volume 16, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-16/number-2/articles/volume16-number2.pdf

Page 5: State variable filter

Volume 16, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-16/number-3/articles/volume16-number3.pdf

Page 20: AD539 Dual analog multiplier

Volume 17, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-17/number-1/articles/volume17-number1.pdf

Page 8: Shielding and Guarding

Volume 17, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-17/number-2/articles/volume17-number2.pdf

Page 11: Guarding

Page 12: Digital FIR Filters

Volume 18, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-18/number-1/articles/volume18-number1.pdf

Page 3: DC Amplifier Noise

Page 11: AD637 RMS to DC Converter

Page 14: Fast, simple Approximation of Functions

Volume 18, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-18/number-2/articles/volume18-number2.pdf

Page 16: AD630 Balanced Modulator / Demodulator

Volume 18, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-18/number-3/articles/volume18-number3.pdf

Page 12: AD639 Trigonometric Function Generator

Volume 19, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-19/number-1/articles/volume19-number1.pdf

Page 3: AD538 Analog Multifunction Chip

Volume 20, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-20/number-2/articles/volume20-number2.pdf

Page 19: LVDT (Linear Variable Differential Transformer)

Volume 21, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-21/number-1/articles/volume21-number1.pdf

Page 10: 2S81 Resolver to Digital Converter

Page 12: AD588 Precision Voltage Reference

Volume 22, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-22/number-2/articles/volume22-number2.pdf

Page 12: AD840 Series

Page 16: AD9300 Video Multiplexer, Chroma Keying

Volume 23, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-23/number-1/articles/volume23-number1.pdf

Page 7: AD689 8.192V Reference

Page 8: AD834 4-Quadrant Multiplier

Volume 23, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-23/number-3/articles/volume23-number3.pdf

Page 3: AD640 Logarithmic Amplifier

Page 7: Grounding

Page 15: AD598 LVDT Conditioner

Page 16: 6S04 Synchro Simulator and Tester

Volume 24, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-24/number-1/articles/volume24-number1.pdf

Page 12: some infos about Synchros and Resolvers

Volume 24, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-24/number-3/articles/volume24-number3.pdf

Page 20: AD734 Multiplier

Page 20: AD688 +-10V Reference

Volume 25, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-25/number-1/articles/volume25-number1.pdf

Page 10: AD9950 Direct Digital Synthesis

Page 23: AD633 Multiplier

Volume 26, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-26/number-1/articles/volume26-number1.pdf

Page 10: AD811 Video Operational Amplifier

Volume 26, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-26/number-2/articles/volume26-number2.pdf

Page 13: OP Amps in Line Driver and Receiver Circuits, Part 1

Volume 27, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-27/number-1/articles/volume27-number1.pdf

Page 14: OP Amps in Line Driver and Receiver Circuits, Part 2

Volume 28, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-28/number-1/articles/volume28-number1.pdf

Page 13: Voltage References

Volume 28, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-28/number-3/articles/volume28-number3.pdf

Page 6: Clamping

Volume 29, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-29/number-1/articles/volume29-number1.pdf

Page 8: Digitally controllable variable Resistors

Page 13: Clamp Amplifiers

Volume 29, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-29/number-3/articles/volume29-number3.pdf

Page 15: AD7805

Volume 30, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-30/number-3/articles/volume30-number3.pdf

Page 20: Current Feedback Amplifiers, Part 1

Volume 30, Number 4: https://www.analog.com/media/en/analog-dialogue/volume-30/number-4/articles/volume30-number4.pdf

Page 20: Current Feedback Amplifiers, Part 2

Volume 31, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-31/number-2/articles/volume31-number2.pdf

Page 19: OP Amps driving capacitive Loads

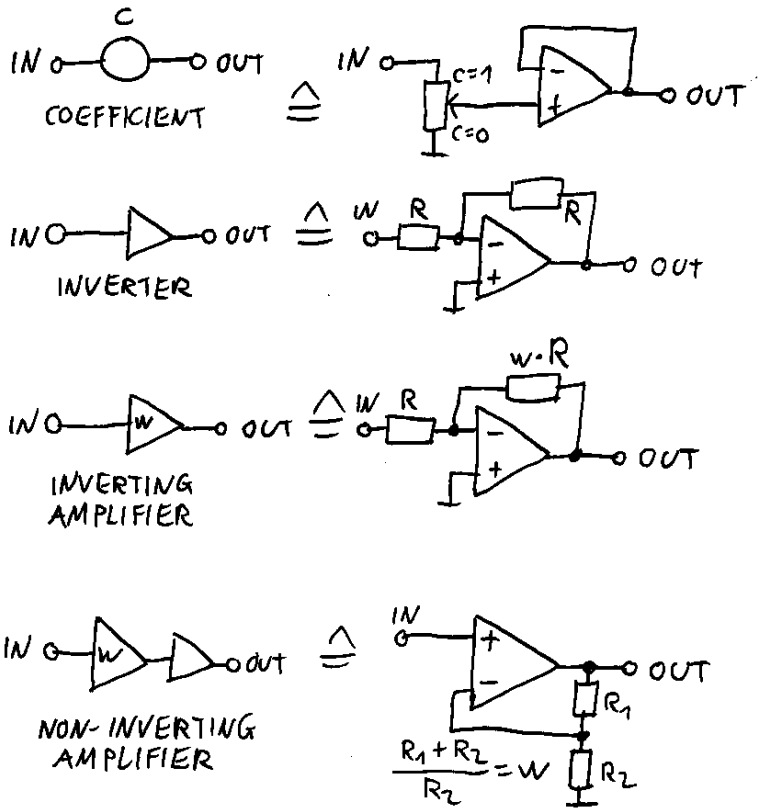
.....

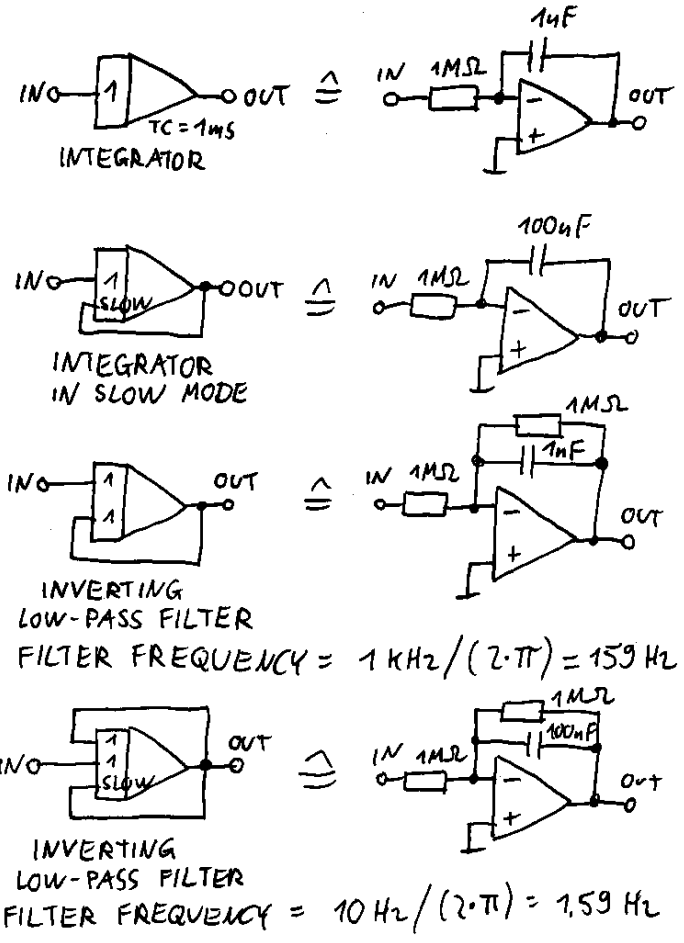
Volume 42, Number 4: https://www.analog.com/media/en/analog-dialogue/volume-42/number-4/articles/volume42-number4.pdf

Page 8: Considering Multipliers, Part 1

2 Components

2.1 Translate from Analog Computing Symbols to OP-Amp Circuits





2.2 Compatibility with +-15 V Signals

All inputs and outputs in THAT are protected against electrostatic discharge with ESD Protection diodes:

Vishay Semiconductors VCAN16A2-03S-E3-08

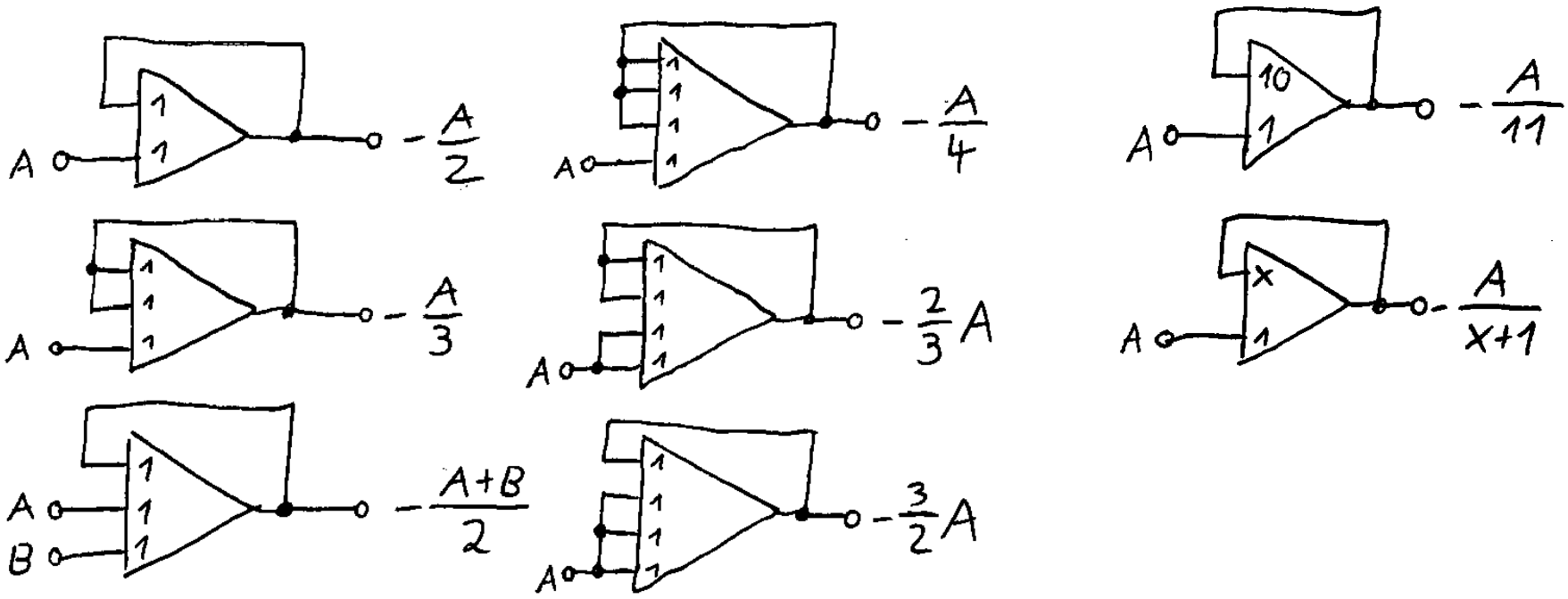
The usable voltage (without breakdown) is up to 16.0V. The reverse breakdown voltage is in the 17.1V to 20.0V range.

Data sheet: https://www.vishay.com/docs/86175/vcan16a2-03s.pdf

This means you can connect +-15 V signals to all inputs, except to the following:

- **All SJ inputs (because they don't have input resistors, and are clamped against the +-12 V supply rails at the inputs of the OP-Amps).**
- **All ">0" and "<0" inputs of the comparators (because they don't have input resistors, and are clamped against the +-12 V supply rails in the analog switches).**

2.3 Summer



Please note that the output signals are different in open amplifiers and summers!

See also: https://the-analog-thing.org/wiki/Summer

2.4 Open Amplifier

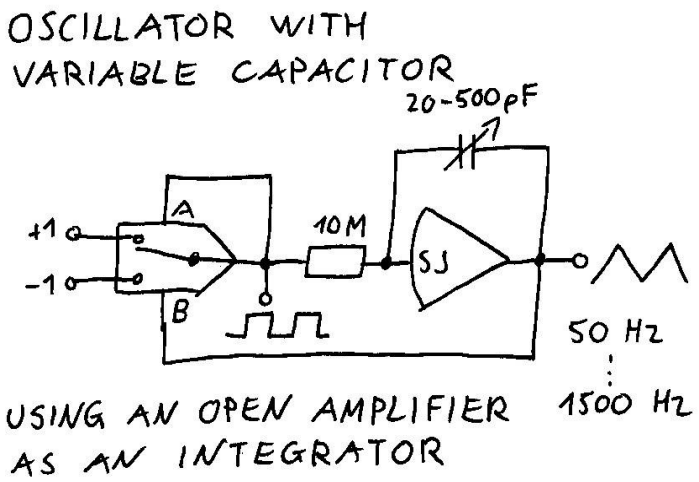
The open amplifier adjusts its output so that the sum of the inputs becomes zero. This does only work if the sign isn't changed in the feedback loop.

An open amplifier can also be used as an integrator. The advantage is that this integrator keeps its value when THAT is in stop mode. The drawback is that you can't set an initial condition.

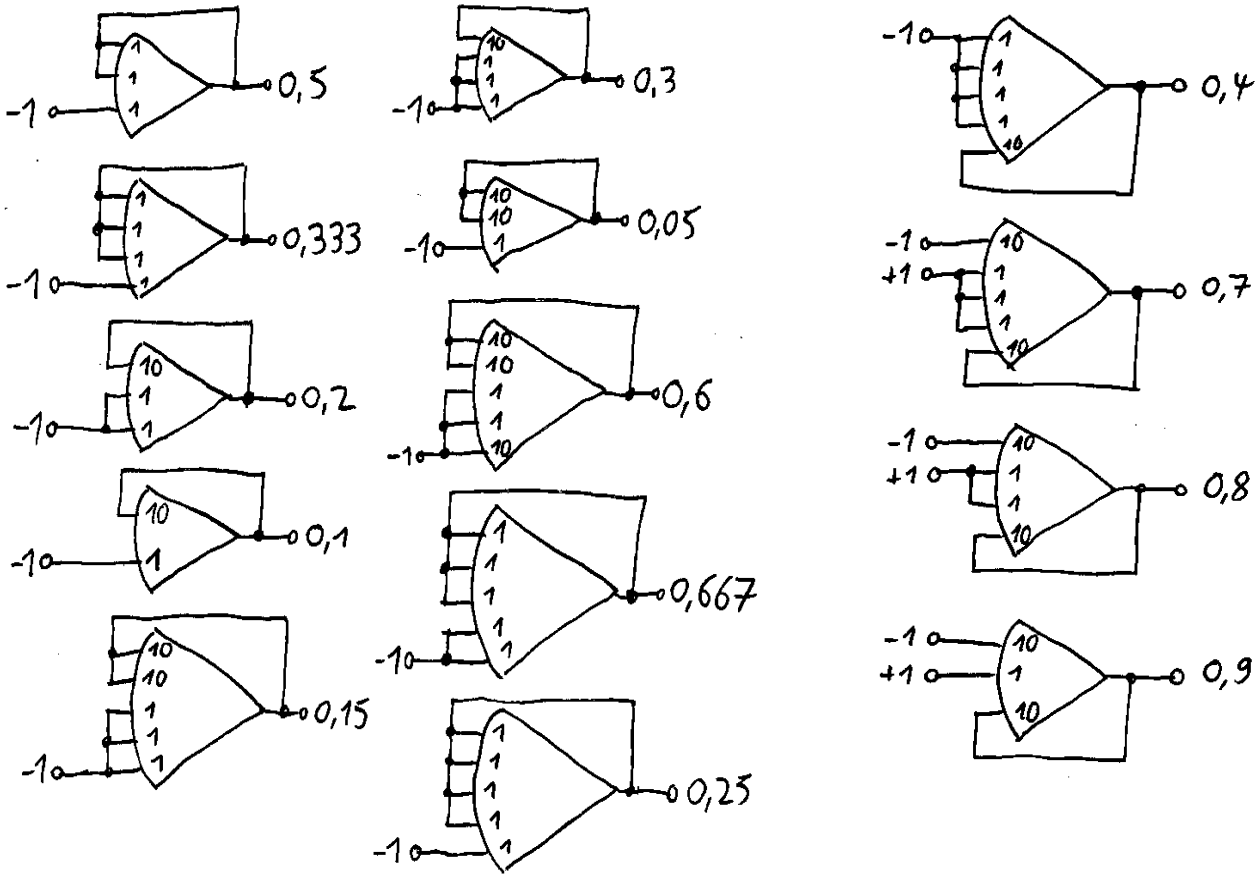
This can also be used if you need an integrator with a longer time scale factor.
1µF · 1MΩ = 1s 10µF · 1MΩ = 10s 10µF · 10MΩ = 100s 10µF · 100MΩ = 1000s
100µF · 100MΩ = 10000s 10µF · 1GΩ = 10000s 3.6µF · 1GΩ = 3600s = 1h 100µF · 36MΩ = 3600s = 1h

100 MΩ resistors are available from Farnell: https://de.farnell.com/te-connectivity/rgp0207chj100m/dicksch-widerstand-100m-5-250mw/dp/2805251

1 GΩ resistors are also available from Farnell: https://de.farnell.com/te-connectivity/rgp0207chk1g0/dickschichtwiderstand-1g-10-250mw/dp/2805261



An open amplifier can also be used for making precision constants, without having to adjust coefficients:

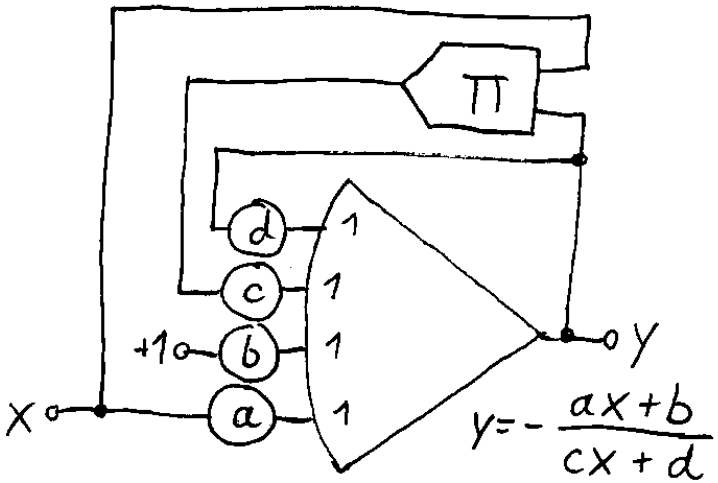


For negative constants, simply swap the polarity of the -1 and +1 inputs.

Please note that the output levels are different in open amplifiers and summers!

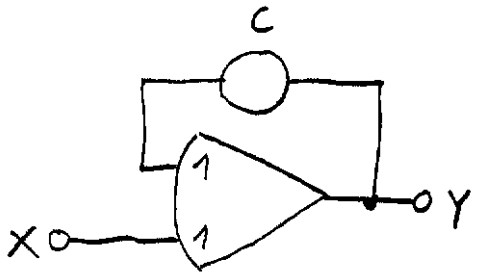
See also https://the-analog-thing.org/wiki/Summer

Function y = -(ax + b) / (cx + d)



$$ax + b + cxy + dy = 0$$

Function y = - x / c



$$y = -\frac{x}{c}$$

2.5 Operational amplifiers with vacuum tubes

https://www.allaboutcircuits.com/textbook/semiconductors/chpt-8/operational-amplifier-models/

https://www.rfcafe.com/references/electronics-world/operational-amplifier-electronics-world-july-1963.htm

12AX7 = ECC83 https://www.tube-town.net/ttstore/roehren/vorstufenroehren/12ax7-ecc83-5751/?language=de

http://www.philbrickarchive.org/

http://www.philbrickarchive.org/k3_series_components.htm

http://www.philbrickarchive.org/k2-w_operational_amplifier_later.htm

http://www.philbrickarchive.org/k2-x_100v_xperimental_opamp.htm

http://www.philbrickarchive.org/k2-xa_operational_amplifier.htm

http://www.philbrickarchive.org/k2-yj_rugged_operational_amplifier.htm

http://www.philbrickarchive.org/sk2-v_schematic_and_applications.htm

http://www.philbrickarchive.org/operational_amplifier.htm

2.6 Integrator

The integration time constant TC in THAT is 1ms, or 100ms in SLOW mode. If you need 10ms time scale factor, you can connect an external 10nF capacitor between OUT and SLOW. The exact value is 9.89nF, so you can select a capacitor with -1.1% tolerance.

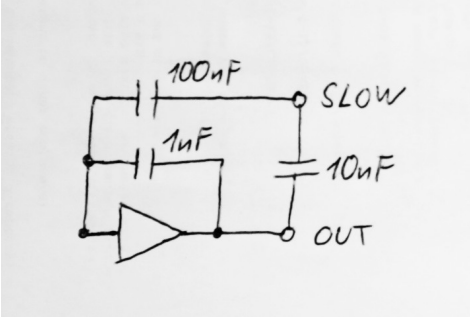
$$OUT(t) = TC \cdot \int IN(t)\, dt \qquad IN(t) = 1 / TC \cdot dOUT(t) / dt$$

All time constants between 1ms and 100ms can be realized with an external capacitor between SLOW and OUT.

The time constant in ms can be calculated as follows: TC = (100 + 101 · C) / (100 + C) where C is the external capacitor in nF.

The external capacitor in nF can be calculated as follows: C = (100 - 100 · TC) / (TC - 101) where TC is the time constant in ms.

Time constant TC	External capacitor between SLOW and OUT
1 ms	open
3 ms	2.04 nF
10 ms	9.89 nF (10 nF with -1.1% tolerance)
30 ms	40.8 nF
100 ms	short



Integrators with capacitors larger than 100nF can be realized with open amplifiers.

Integrators with time constants larger than 100ms can also be realized by using larger input resistors (10 MΩ, 100 MΩ, 1000 MΩ).

The output of the integrator is inverting. The initial condition is set before the inverter.
That means if you set IC to -1, then the output will initially be at +1.

What about the integrator's time scale factor k_0? A definition for k_0 is given in the book "Analogrechnen" by Giloi and Lauber (on the page before page 1, and on pages 65 and 67): $k_0 = 1 / (R \cdot C)$, which means the unit of k_0 is [1/s]. k_0 is the inverse of TC: $k_0 = 1 / TC$

See also: https://the-analog-thing.org/wiki/Integrator

How fast does the integrator settle to the initial voltage, or with other words: What's the minimum duration of the initial condition?

The time constant is τ = 100nF · 10kΩ = 1ms and the voltage in the capacitor follows an exponential function:

$$U = U_0 \cdot e^{(-t/\tau)}$$

As the worst case U_0 can be 2 machine units, if the capacitor must be discharged from +1 MU to -1MU.

IC Duration	Voltage U in capacitor, for U_0 = 2 = Error in machine units
1ms	0.736
2ms	0.271
3ms	0.100
4ms	0.037
5ms	0.013
6ms	0.0050
8ms	0.0007
10ms	0.0001

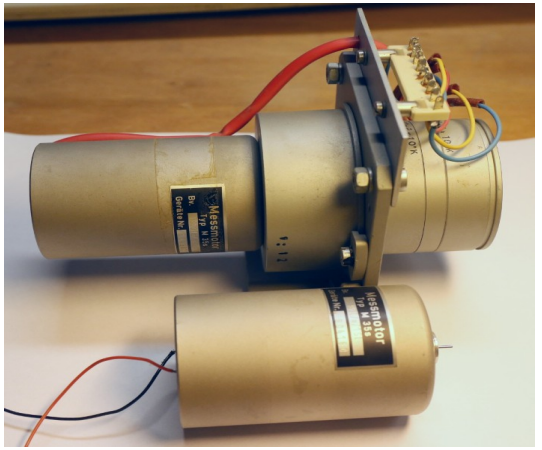
2.7 Electromechanical Integrator

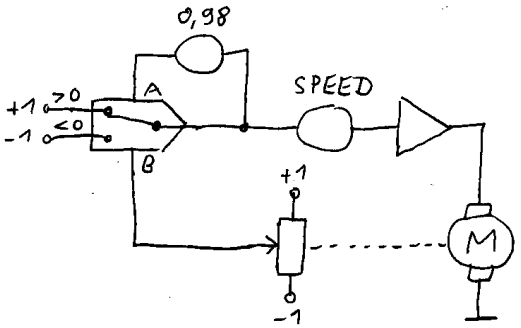
These are motors from an electromechanical integrator, probably from the 1960s or 1970s. One of the motors has a 1:12 gearbox and a precision stereo potentiometer, 2640 Ω and 195 Ω.
Motor without gearbox: "Messmotor Typ M 35s, Bv. 900/a/1St". Motor with gearbox: "Messmotor Typ M 35s, Bv. 1800/a/1St".

The motor current is well below 1mA, so it can directly be connected to a THAT output.

The maximum voltage for the "M 35s Bv. 1800/a/1St" motor is 14V.

The motor begins to run at 0.15V = 0.015 machine units, and the time constant is about 3.25s (potentiometer from end to end, with 10V motor voltage).





My measurement results:

Motor Voltage M 35s 1800/a/1St		Duration t 270° end to end	t · MU This should be a constant	Notes
V	MU			
0.25	0.025	228s	5.7s	Speed is significantly irregular
0.5	0.050	85.2s	4.26s	Speed is slightly irregular
1	0.100	36.8s	3.68s	
1.5	0.150	23.3s	3.49s	
2	0.200	17.2s	3.44s	
2.5	0.250	13.6s	3.37s	
3	0.300	11.3s	3.39s	
4	0.400	8.35s	3.34s	
5	0.500	6.62s	3.31s	
6	0.600	5.49s	3.29s	
7	0.700	4.70s	3.29s	
8	0.800	4.09s	3.27s	
9	0.900	3.63s	3.27s	
10	1.000	3.25s	3.25s	

There are some details about this motor in this document on page 4: https://docplayer.org/204259855-Dehnungsmessanlage-fuer-genaue-leistungsmessungen-an-zapfwellenbetriebenen-landmaschinen-mit-hilfe-von-in-tegratoren.html

The same information is also in this document on page 133: https://mediatum.ub.tum.de/doc/1518280/1518280.pdf

Tafel 1: Auszug aus der Auswahl von Meßmotoren

Meßbereich V	Stromaufnahme bei Nennspannung mA	Meßmotor M 35s Bv	Solldrehzahl bei Meßbereichendwert U/min	Fehler max. % bezogen auf anliegenden Wert bei U_N 100 %	10 %
0 . . . 1.75	1.5	225/a			
0 . . . 3.5	0.9	450/a		ca.	ca.
0 . . . 7	0.5	900/a	240	+ 0.15	— 1.5
0 . . . 14	0.3	1800/a			
0 . . . 24	0.2	3000/a			

[11] Alle Angaben über den integrierenden Impulsgeber wurden einer Druckschrift der Firma Fernsteuergeräte ZACHARIÄ-OELSCH-MAIER in Berlin 47, entnommen

Oelsch, Kurt, Ing. grad., 1000 Berlin 37, Glockenstr. 4, * 15.4.1919
PershGes: Fernsteuergeräte, Zachariä-Oelsch-Meier, 1000 Berlin 47, Jahnstr. 68 - 72
Fernsteuergeräte KG Zachariä-Oelsch-Meier, 6148 Heppenheim (Bergstr), Weiherhausstraße

That means at 10V (1 machine unit) the speed of the "M 35s Bv. 1800/a/1St" motor is 240 / 14 · 10 = 171.4 revolutions per minute, or 2.857 revolutions per second.

After the 1:12 gearbox the speed is 0.238 revolutions per second, or 4.2 seconds for 360° angle or 3.15 seconds for the 270° sector of the potentiometer. It seems my motor is running about 3% too slow.

The company does still exist! https://www.fsg-sensors.de/about

2.8 Differentiator

See chapter 5.14 in Bernd's book "Analog and Hybrid Computer Programming".

Or use an open amplifier with an integrator in the feedback loop.

2.9 Comparator

The comparators in THAT have about 0.022V hysteresis (about 0.002 machine units).

https://github.com/anabrid/the-analog-thing/blob/main/ANATHING_OSH/ANATHING-BASE%20-%20Schematic4.pdf

See also https://the-analog-thing.org/wiki/Comparator

Comparator outputs are switching very fast, which means these signals contain high frequencies. In order to avoid crosstalk to other signals, it's a good idea to keep the cables at the comparator's output as short as possible.

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 154ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

2.10 AD633 Multiplier

In THAT the AD633JR chip is used. Because this chip is specified for +-15V supply voltage, but used in THAT with +-12V supply, the inputs are divided by 2 and the output is multiplied by 4.

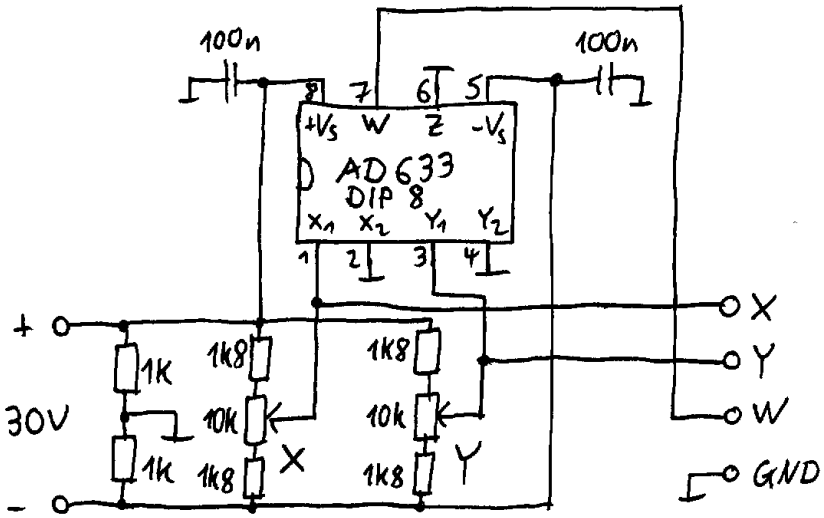
Data sheet: https://www.analog.com/media/en/technical-documentation/data-sheets/AD633.pdf

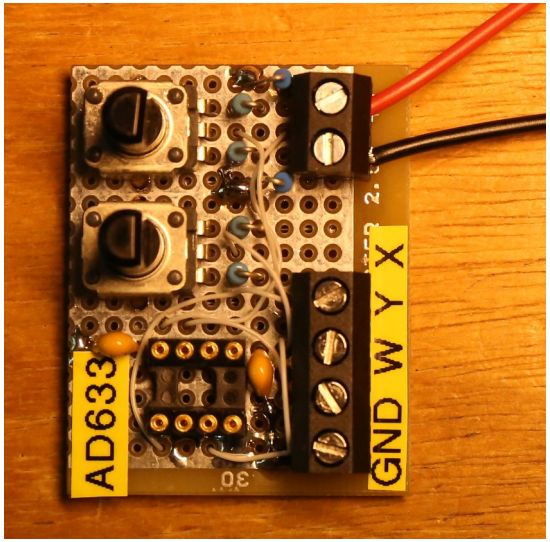
Transfer function in Volts: out = x · y / 10V

Transfer function in machine units: OUT = X · Y

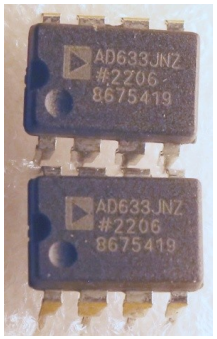
with X = x / 10V
Y = y / 10V
OUT = out / 10V

If you want to build your own circuits with the AD633 multiplier, please note that Ebay and Aliexpress are full of counterfeit AD633 chips. Most likely all AD633 chips for less than $10 are counterfeit. If you find a genuine AD633 for less than $10, please let me know. The counterfeit chips do not work at all. It's not even possible to set the input voltage to the desired level, because pins 1 and 3 aren't high impedance. If you force the input voltage to the desired level, the chip becomes hot. You can use this circuit for testing if AD633 is genuine or counterfeit. Pinout is for DIP8. Pinout for SO8 is different!



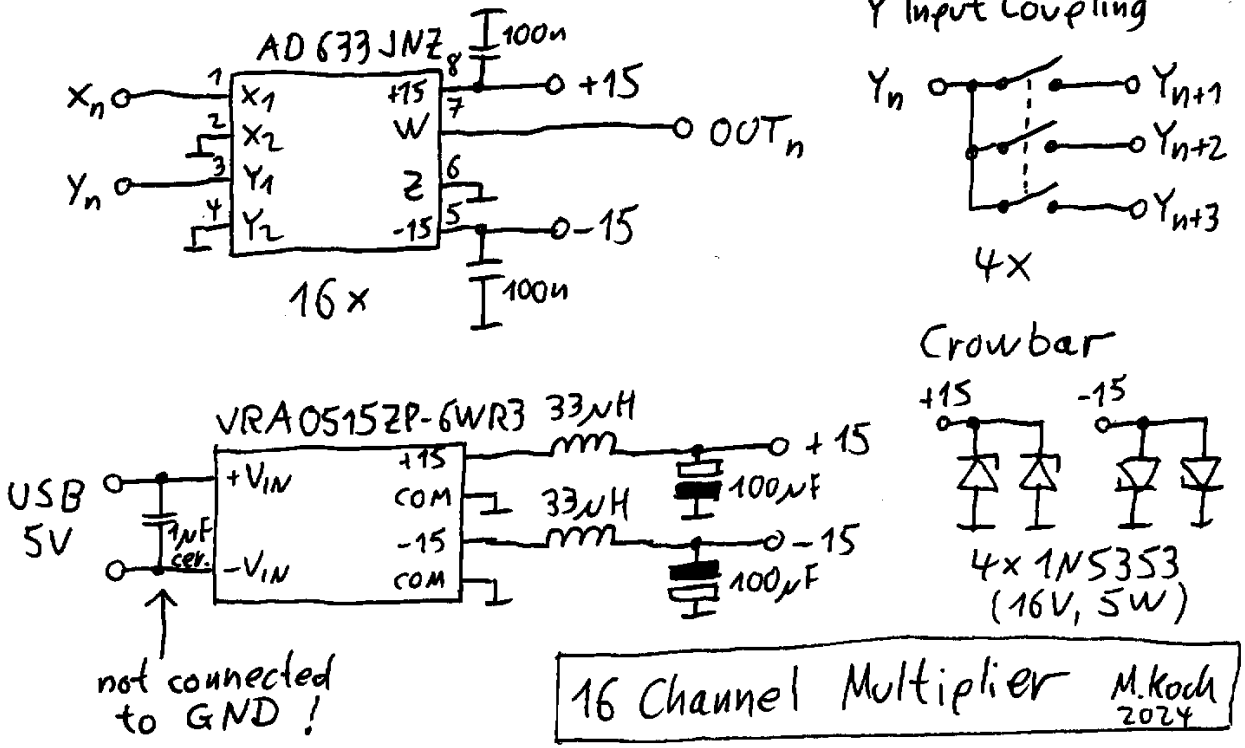


AD633 chips with these date codes were tested as fake:

Text on chip	Seller
▷AD633JNZ #1828 3782755	https://www.ebay.de/str/survy2014
▷AD633JNZ #2206 8675419 	https://www.ebay.de/str/sensecore Reply from seller: "Hello. Our product is AD633JN, this is our domestic chip, that's why the price is lower, the original AD633 will be much more expensive. We can refund you 20EUR as compensation. would you like to keep the item?it can save your time and engery. Pls consider,thanks! Waiting for your reply. Your sincerely" The same seller has at least two other accounts on Ebay: https://www.ebay.de/str/diyassistant (modul_technik) https://www.ebay.de/str/richdede0 (worldchips)
▷AD633JN #2114 2618073	https://www.ebay.de/str/roboterbausatz I bought two chips. One doesn't have high impedance at pins 1 and 3, so that's impossible to set the input signals to the desired level. The other one doesn't contain any silicon. Infinite resistance between any pair of pins.
▷AD633JNZ #2218 3860725 	https://www.ebay.com/str/huayicomponents Two counterfeit chips with identical data codes, but containing two different unknown chips.

AD633 chips with these date codes were tested ok:

Text on chip	Seller
▷AD633JN #2343 6439819	https://www.reichelt.de
▷AD633JN #2430 6669435	https://www.mouser.de



16 Channel Multiplier M.Koch 2024

2.11 AD538 Multiplier

$V_O = V_Y \cdot (V_Z / V_X)^M$ https://www.analog.com/media/en/technical-documentation/data-sheets/AD538.pdf

For the circuit in figure 15 in the data sheet, a special resistor 1090 Ω +3300ppm/K (not exact, see below) is required. 1%/K = 10000 ppm/K

This temperatuire coefficient is in more detail explained in this article (page 3):

https://www.analog.com/media/en/analog-dialogue/volume-19/number-1/articles/volume19-number1.pdf

The required temperature coefficient is 3333 ppm/K or 0.333 %/K. This is simply because the output voltage of the log section is proportional to the absolute temperature. It we assume T=300K (+26.85°C), then the temperature coefficient is 1/300 = 3333ppm.

In my opinion there is an error in the data sheet. A 90.9 Ω resistor in series with 1000 Ω +3500 ppm/K does not give 1090 Ω +3333 ppm/K, but instead +3208 ppm/K. See also https://ez.analog.com/amplifiers/operational-amplifiers/f/q-a/572296/temperature-coefficient-in-ad538

Platin PT500 3850 ppm/K https://www.reichelt.de/platin-temp-sensor-22-x-4-0-mm-500-ohm-pcb-2240-pt500-p151245.html

Platin PT1000 3850 ppm/K https://www.reichelt.de/platin-temperatursensor-pt1000-klasse-a-f0-15--ertd2-pt-1000-a-p290780.html

KTY81-122 1000 Ω 0.79%/K = 7900 ppm/K https://www.reichelt.de/temperatursensor-ptc-1-kohm-55-150-c-kty-81-122-p9597.html

KTY 81-222 2000 Ω 0.79%/K = 7900 ppm/K https://www.reichelt.de/temperatursensor-ptc-2-kohm-55-150-c-kty-81-222-p9601.html

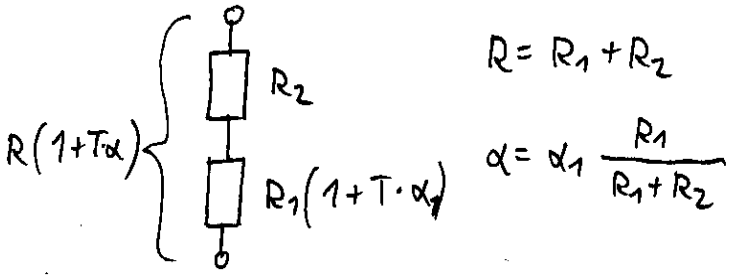
KTY 84-130 1000 Ω 0.62%/K = 6200 ppm/K https://www.reichelt.de/temperatursensor-40-300-c-kty-84-130-p9604.html

AN-304 Application Note: https://www.analog.com/media/en/technical-documentation/application-notes/510571309728142264494067 67AN304.pdf

Considering Multipliers (Part 1): https://www.analog.com/en/analog-dialogue/articles/considering-multipliers-part-1.html

Analog Computation and Signal Processing: https://www.analog.com/media/en/training-seminars/design-handbooks/Linear-Design-Seminar-1987/Section5.pdf

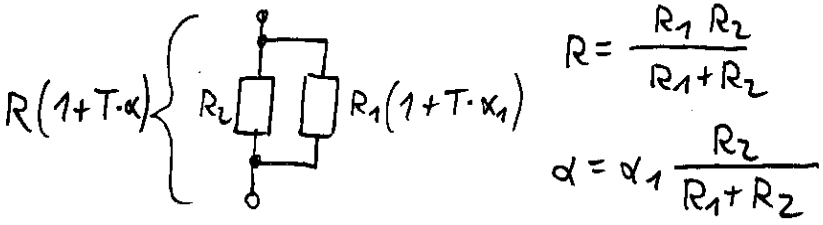
These are the formulas for the resulting temperature coefficient alpha when two resistors are connected in series or parallel. R1 has temperature coefficient alpha1 and R2 doesn't have a temperature coefficient.



Calculate the unknown series resistor R2, if the resulting temperature coefficient alpha is known:
R2 = R1 · (alpha1 / alpha - 1)

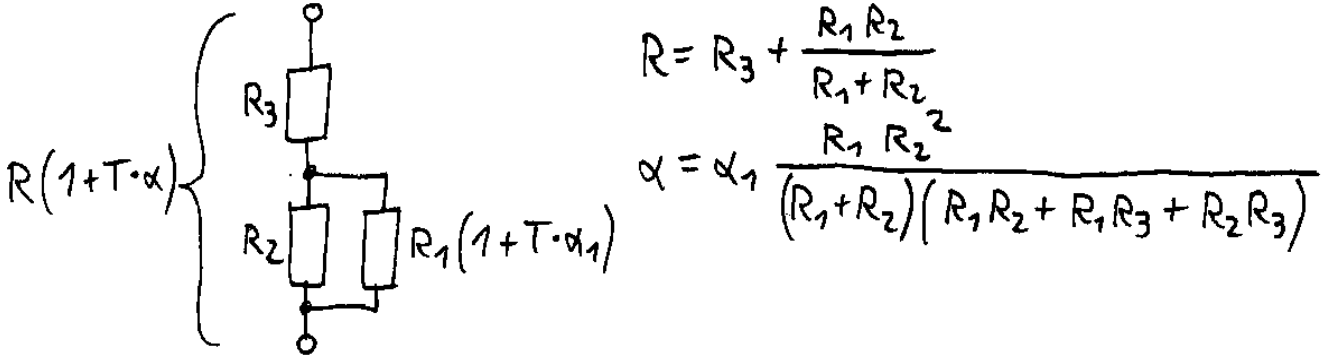
Examples for AD538:
For PTC 1000 Ω +3500 ppm/K and alpha = +3333 ppm/K: R2 = 50.1 Ω
For PT1000 (alpha = +3850 ppm/K) and alpha = +3333 ppm/K: R2 = 155 Ω

Calculate the resulting temperature coefficient alpha if both resistors have temperature coefficients alpha1 and alpha2:
Series: alpha = (alpha1 · R1 + alpha2 · R2) / (R1 + R2)
Parallel: alpha = (alpha1 · R2 + alpha2 · R1) / (R1 + R2)



These are the formulas for the resulting temperature coefficient alpha when three resistors are connected. R1 has temperature coefficient alpha1 and R2 and R3 don't have temperature coefficients.

I don't know if it's analytically solvable for R2 and R3, if R, R1, alpha and alpha1 are given. If you find a solution, please let me know.



2.12 Diodes

The diodes in THAT are BAS170W Schottky diodes.

Typical forward voltage: 375mV at 1mA, 705mV at 10mA

Reverse current: < 0.1uA at 50V

Data sheet: https://www.mouser.de/datasheet/2/196/bas70_bas170series_2014-85076.pdf

https://www.infineon.com/dgdl/Infineon-BAS70series-DataSheet-v01_00-EN.pdf?fileId=5546d4626d66c2b1016d73f7ecf220d6

This reverse current may produce significant errors if the diode is connected to the summing junction of an integrator with a very long time constant.

1N4148:

Maximim forward voltage: < 1 V at 10mA (but typical forward voltage is smaller)

Reverse current: < 0.025uA at 20V

Data sheets: https://www.vishay.com/docs/81857/1n4148.pdf https://www.mouser.de/datasheet/2/308/1N914_D-2309448.pdf

BAS34:

70V, 200mA

Maximim forward voltage: < 1 V at 100mA (but typical forward voltage is smaller)

Reverse current: typical < 0.5nA at 30V Ultra small reverse current

Data sheet: https://www.vishay.com/docs/85541/bas33.pdf

The Z-Diodes in THAT are BZT52HC10. The Zener voltage is 10V +-0.5V.

The dynamic resistance is 70Ω at 1mA, or 15Ω at 5mA. The forward voltage is unspecified.

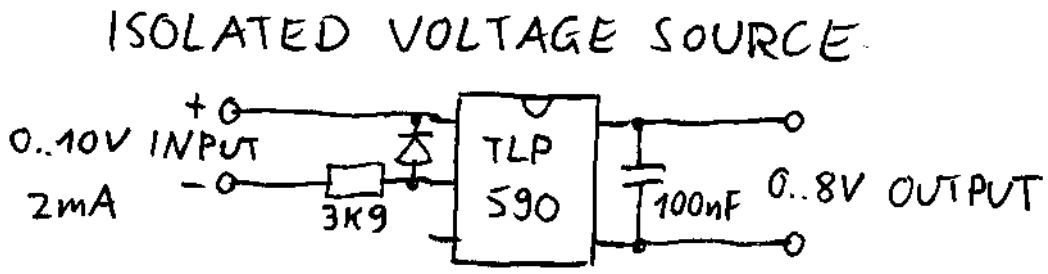
Data sheet: https://www.vishay.com/docs/86342/bzt52_series.pdf

Idealized diode:

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 156ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

2.13 Isolated Voltage Source

Using the TLP590 photodiode array coupler, which has 16 photodiodes in series.



2.14 Cable Length

Rule of thumb: Cable length isn't critical for most connections, but for comparator outputs and summing junction inputs the cables should be as short as possible. Regarding crosstalk, the worst case is capacitive coupling from a comparator output to a summing junction input.

2.15 Analog Panel Meters

As an analog output device for THAT, it's best to use +-1mA meters with a 9.85kΩ series resistor (10k || 680k), so that 1 mA is equivalent to 1 machine unit (10V).

+-1mA meters: https://de.aliexpress.com/item/1005003385410991.html https://www.ebay.com/itm/184533810389 https://de.aliexpress.com/item/1005007289324726.html https://de.aliexpress.com/item/1005007311136957.html

+-10V meter: https://de.aliexpress.com/item/1005005650339550.html https://de.aliexpress.com/item/1005007098797437.html

+-10V class 1.5 meter: https://de.aliexpress.com/item/1005005655298105.html

+-100µA meters: https://de.aliexpress.com/item/1005005650561266.html https://de.aliexpress.com/item/1005005625158185 https://de.aliexpress.com/item/1005004521461316

+-100µA class 1.5 meter: https://de.aliexpress.com/item/1005003467415360.html

SO-45 100mV meter: https://de.aliexpress.com/item/1005004347234374.html

Type 44C2 is DC meter, class 1.5, 100mm x 80mm

Type 85C1 is DC meter, class 2.5, 64mm x 56mm

Type 85L1 is AC meter, class 2.5, 64mm x 56mm

Type 91C4 is DC meter, class 2.5, 45mm x 45mm, most of these instruments don't have a screw for zero adjustment, you better use SO-45!

Type 91L4 is AC meter, class 2.5, 45mm x 45mm, most of these instruments don't have a screw for zero adjustment, you better use SO-45!

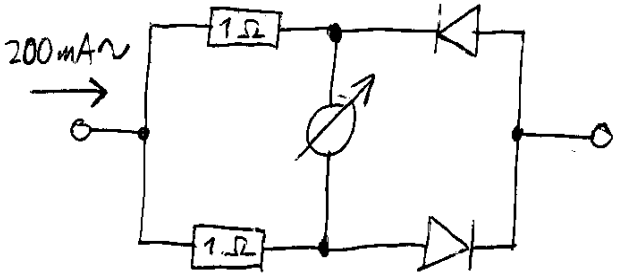
Type SO-45 is AC or DC meter, class 2.5, flange 47mm x 47mm, round 45mm. Available both as *Moving coil meter* which has a linear scale and as *Moving iron meter* which is for AC and DC and has a nonlinear scale.

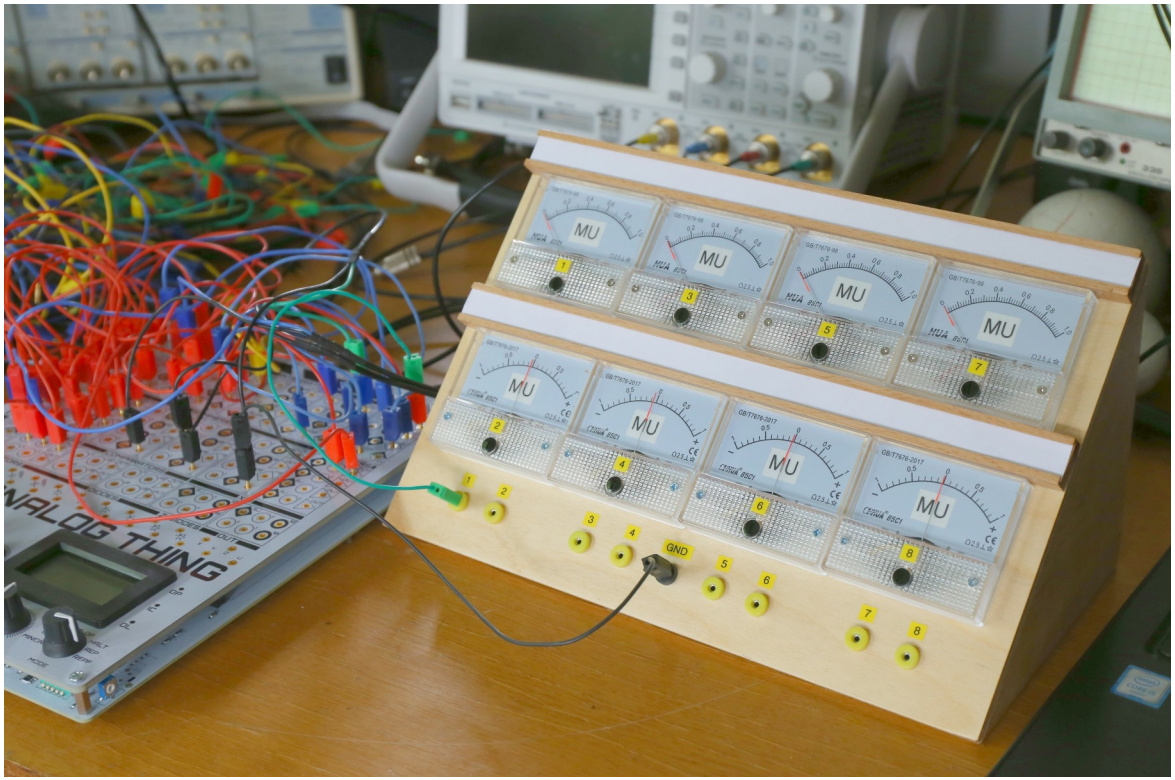
Internal rectifier circuit in 200mA AC meter:

The advantage is that it has a linear scale.

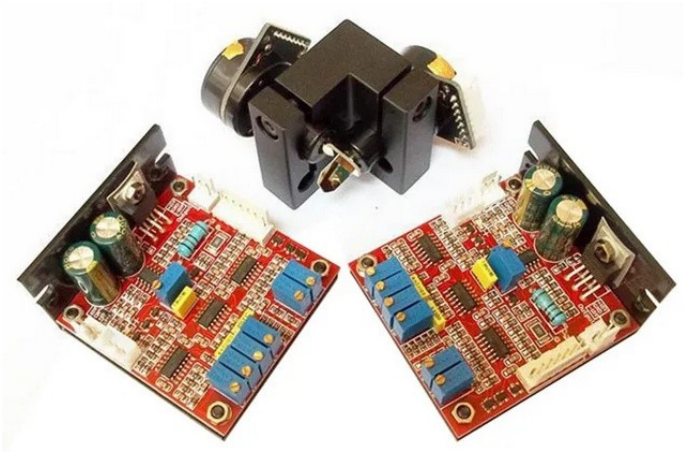
The disadvantage is 0.7V voltage drop in the measuring circuit.

The diodes are 1N4007, but can be replaced by Schottky diodes for lower voltage drop.





2.16 XY Projector with RGB Laser



Deflection angle	Operating voltage	Speed@ Mirrors size
20° optical deflection	+/-15V	18Kpps @ 7*11*0.6
15° optical deflection	+/-15V	25Kpps @ 7*11*0.6
5° optical deflection	+/-15V	30Kpps @ 7*11*0.6

Power input			
XH-3 Connector pins	Description	Remark	Cable color
3	+VCC	+15V/1.0A	RED,24AWG
2	GND		BLACK,24AWG
1	-VEE	-15V/0.6A	WHITE,24AWG
Signal Input			
3	Control signal Y+	-5V~+10V analog signal	
2	S-GND	Ground	
1	Control signal -	-5V~+10V analog signal	

Laser Scanning Galvanometer Type number: B0C7VC1Q16 ?

https://de.aliexpress.com/item/1005006211863006.html
https://de.aliexpress.com/item/32993951454.html
https://www.amazon.de/400-700nm-Speed-Digital-Scanner-Galvanometer/dp/B0C7VC1Q16
https://www.ebay.co.uk/itm/125967153446
https://www.pinterest.de/pin/646125877780970582/
https://de.aliexpress.com/item/33057019195.html (this is a different board)

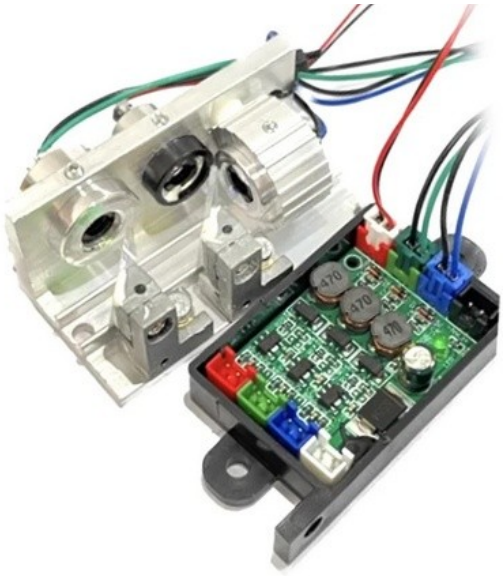
Supply voltage: +-15 V
Peak current: 1.5 A
Scanning angle: +-15° or +-30°

There is no manual available. I tried hard to find a manual, but none of the above sellers is able to provide a manual. It's unclear how to set the potentiometers.

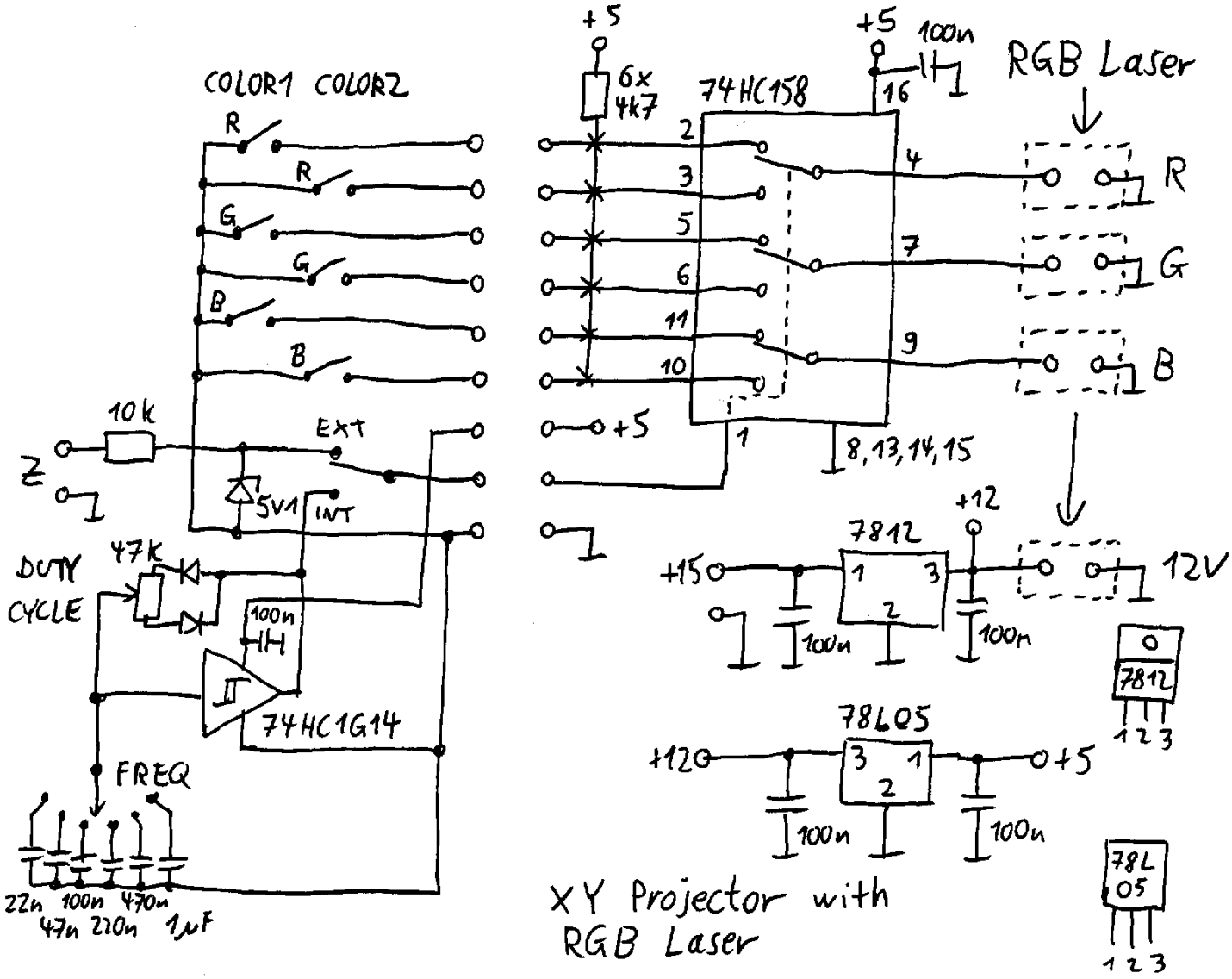
This seller provides a few more informations: https://www.thanksbuyer.com/20kpps-laser-scanning-galvo-scanner-ilda-closed-loop-max-30kpps-laser-parts-59000

Potentiometers:
PS Position scale (increases or decrease input sensitivity of the computer, DO NOT change it)
LIN Zero offset (electrical offset of the driver, adjusted only in factory)
HFD High frequency damping (correct undershoot)
LFD Low frequency damping (correct overshoot)
SG Servo gain (power of the feedback signal for internal PID controller
IS Input scale (adjusted only in factory)
The purpose of the 7th potentiometer is unknown.

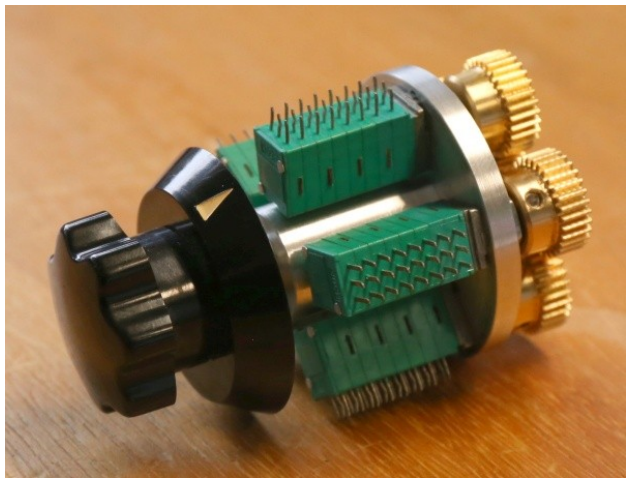
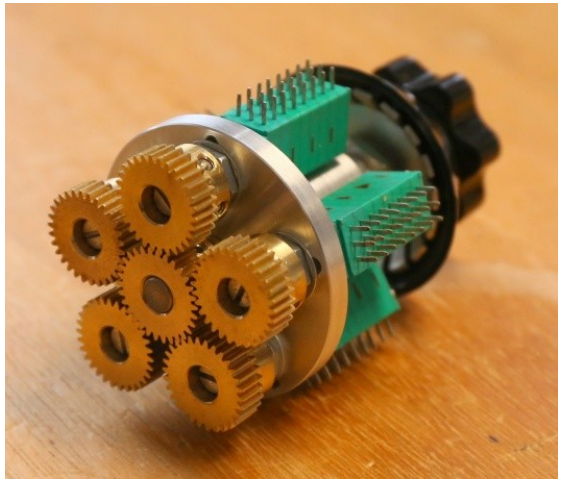
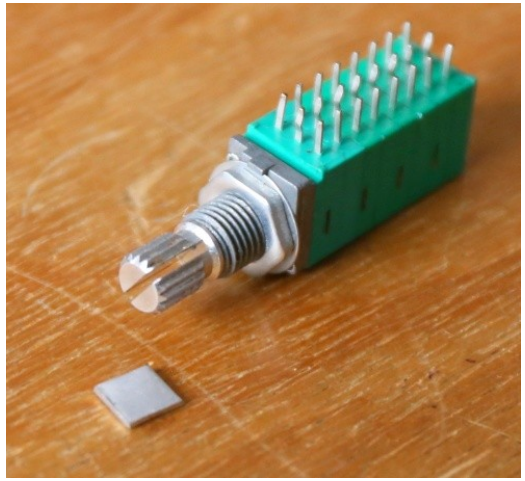
Warning: The pinout of the input connector (left image) is wrong. Pin 1 goes to the non-inverting input, pin 2 goes to the inverting input of TL084C and pin 3 is GND. The output of this OP-Amp goes to the potentiometer in the bottom left corner, which is obviously the input sensitivity adjustment.
If the inverting input is not used, it must be connected to GND.
If the non-inverting input is not used, it can be connected to GND or left unconnected.

	**200mW RGB Laser Module** **https://de.aliexpress.com/item/1005006193483485.html** **https://www.ebay.de/itm/185889792621** **Supply voltage: 12 V** **Current: 0.28 A (measured) 0.8 A or 1.5 A (specification from manufacturer)** **Modulation: 0 – 20 kHz** **The 3 inputs are 5V level active-high, with internal pull-up resistor.** **Check the cables. In my module the green and black wires were swapped in the connector!**
	**Switching power supply D75 +-15V 2.5A** **https://de.aliexpress.com/item/671467231.html**

This is the circuit for controlling the color of the RGB laser.
Two colors can be defined with 6 switches.
An external Z input can be used for switching between these two colors.
Or an internal signal with adjustable frequency and duty cycle can be used.



2.17 Multi-Channel Coefficients

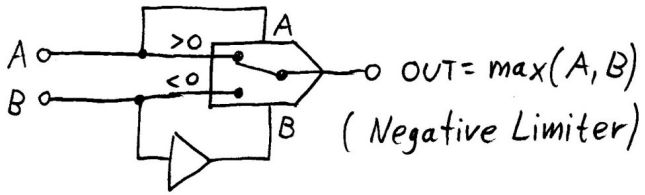




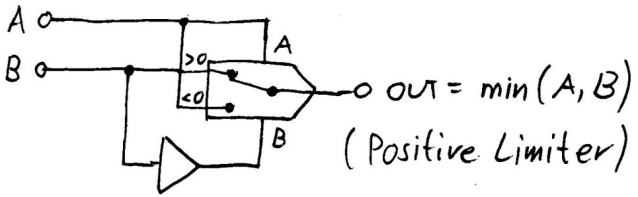
Gearwheels module 0.5, 30 teeth, 6mm bore:
https://de.aliexpress.com/item/1005006753464367.html

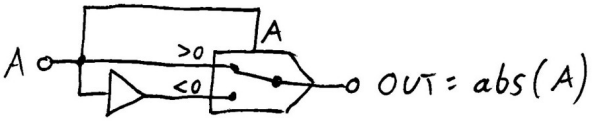
8-channel potentiometers: https://de.aliexpress.com/item/1005005949472081.html

3 Functions

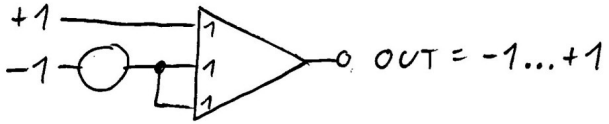
3.1 Helper Functions

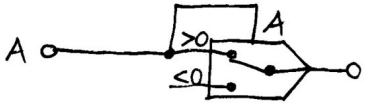






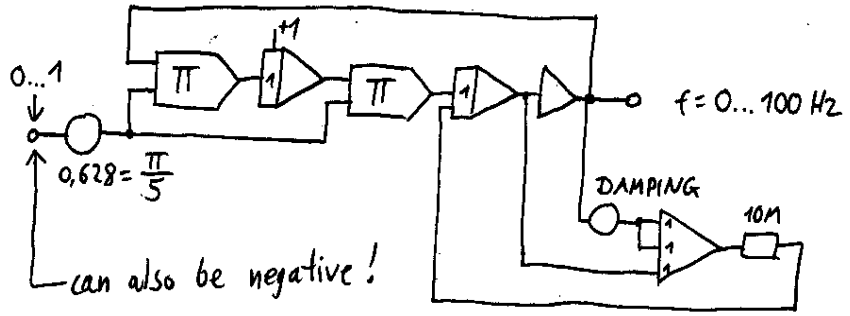
Make an output adjustable from -1 to +1:



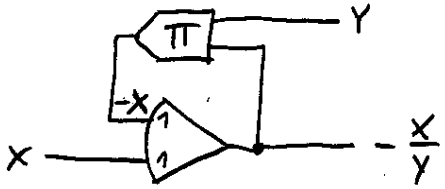


OUT = A if A≥0
OUT = 0 if A≤0

SINE GENERATOR

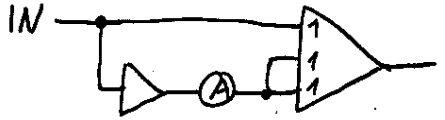


DIVIDER

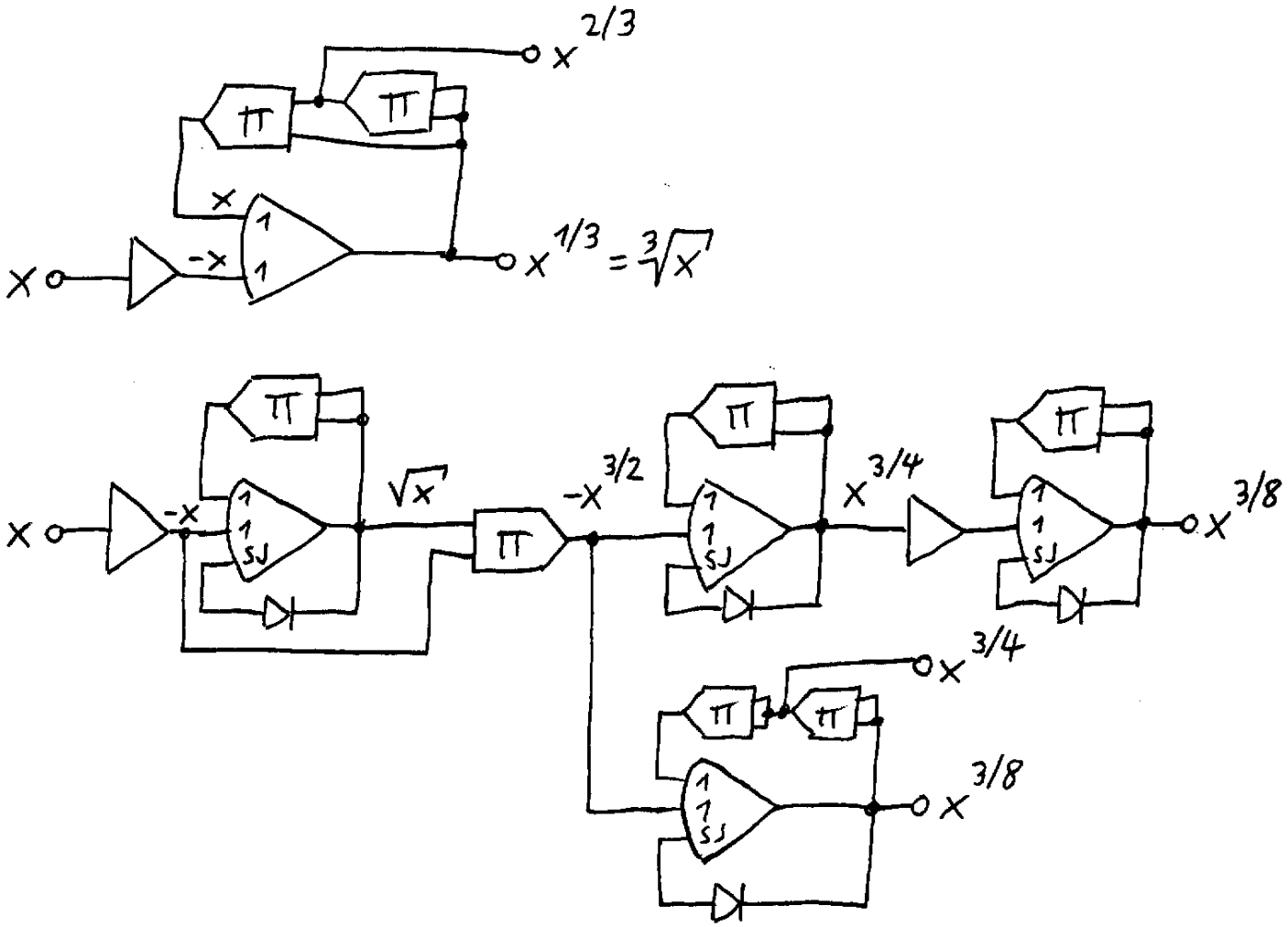


Y ≥ 0

Multiply a variable by a factor in [-1...+1] range:



3.2 Fractional Exponents



3.3 Full Wave Rectifier / Absolute Value

U. Tietze, Ch. Schenk: Halbleiter-Schaltungstechnik, 9. Auflage, page 868

U. Tietze, Ch. Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 1204

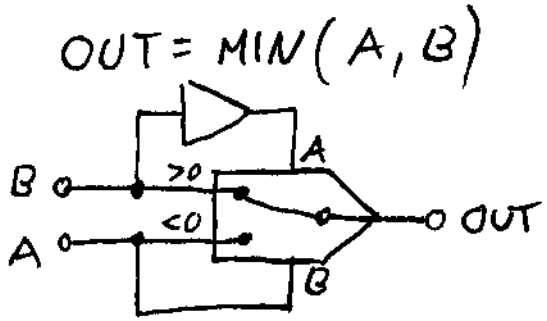
Paul Horowitz, Winfield Hill: The Art of Electronics, Second Edition, page 221

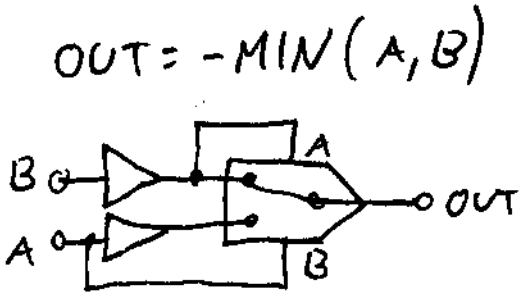
Paul Horowitz, Winfield Hill: The Art of Electronics, Third Edition, page 257

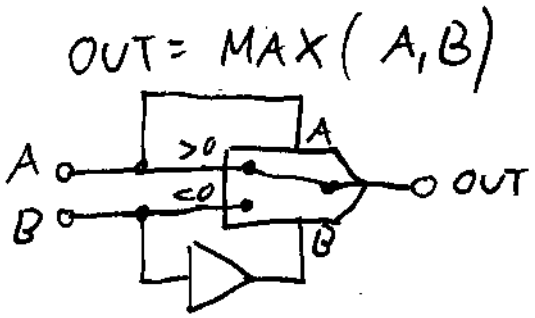
A circuit for absolute value without using a comparator is shown here:

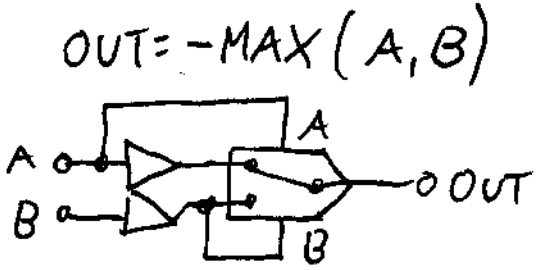
https://the-analog-thing.org/docs/dirhtml/rst/applications/nonlinear_chaos/alpaca_8/

3.4 Minimum and Maximum Functions







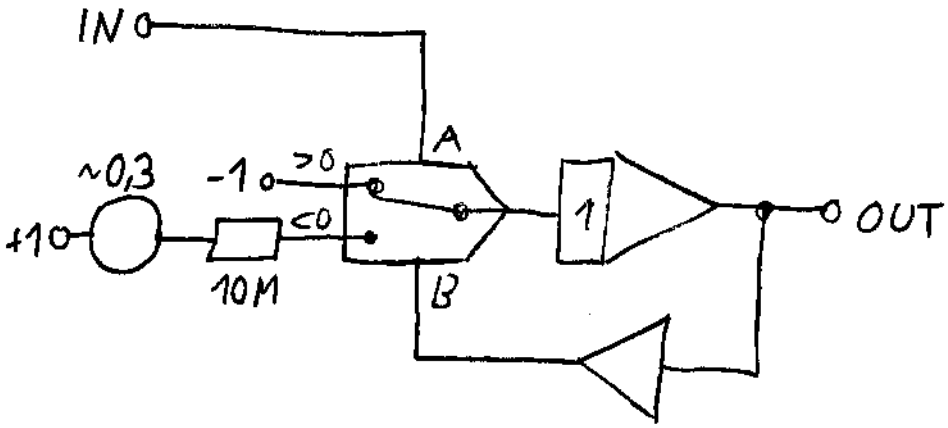


3.5 Measure the Peak Value of a Signal

This circuit compares the input signal with the output signal.

If output < input, the integrator gets -1 at its input, so that the output rises fast.

If output > input, the integrator gets a small positive signal at its input, so that the output falls slowly.



3.6 Limiting the Output of a Summer or Integrator

This works without comparators. When the limits are disabled, the output is not affected at all.

For summers the 2.2nF capacitor is required to remove oscillations. It must be so big because we have two amplifiers in the feedback loop.

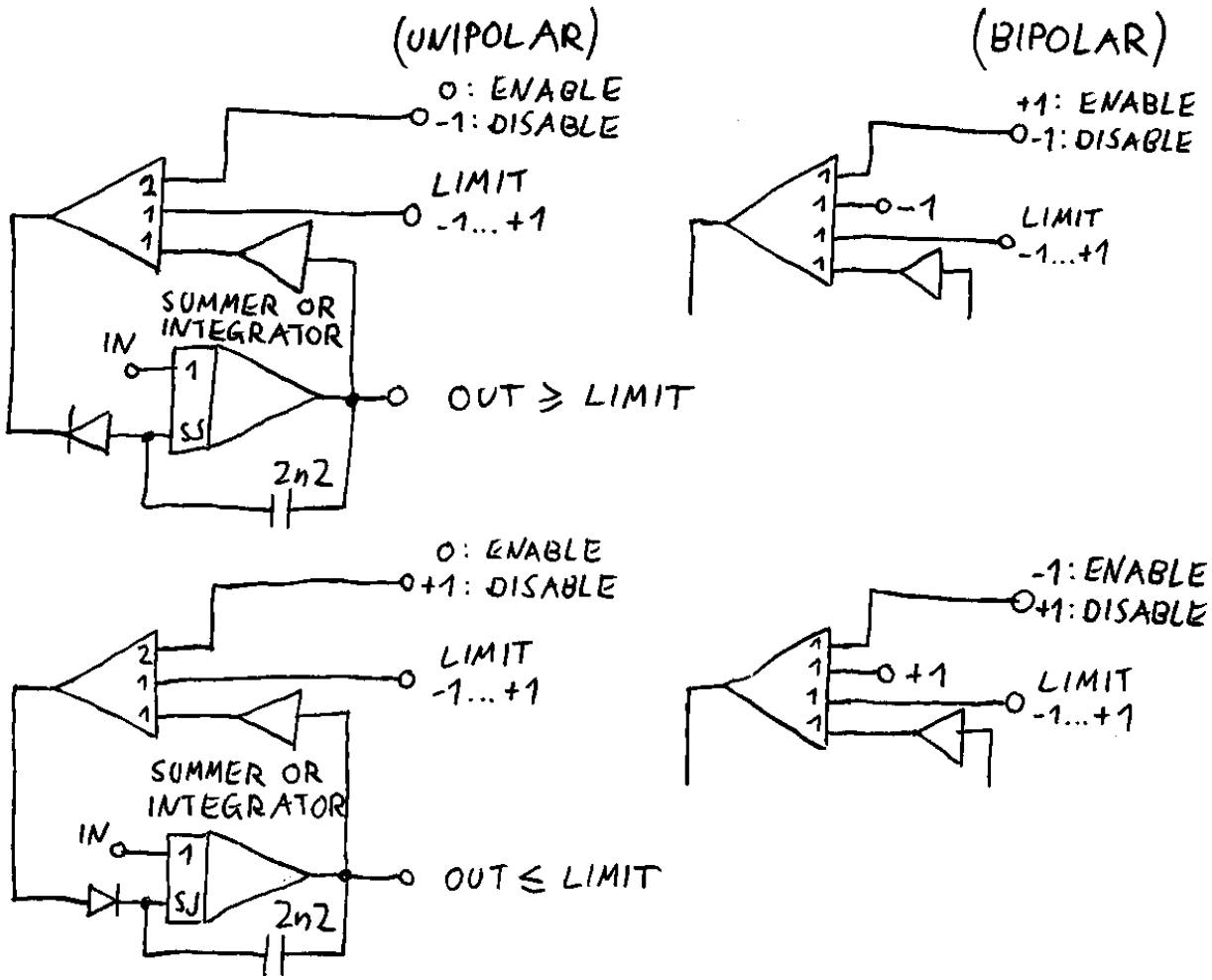
For integrators the 2.2nF capacitor is not required.

The only difference between the two circuits is the polarity of the diode and the polarity of the disable signal.

The enable / disable input can be unipolar (left side) or bipolar (right side).

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 143ff

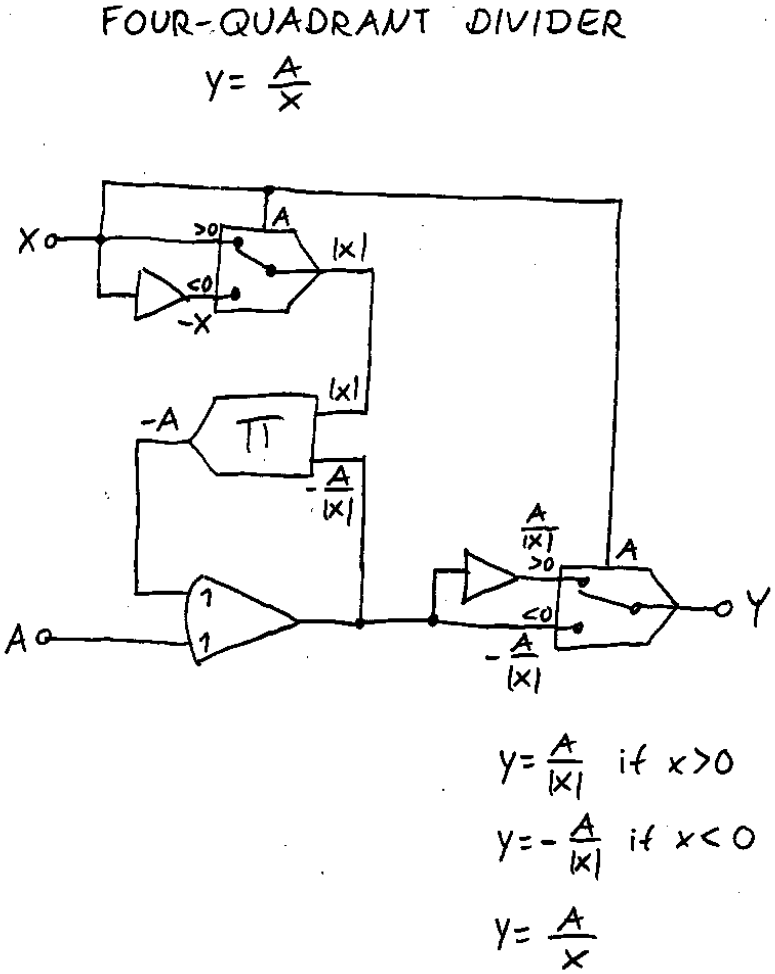
Motion between end-stops: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 173ff



https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

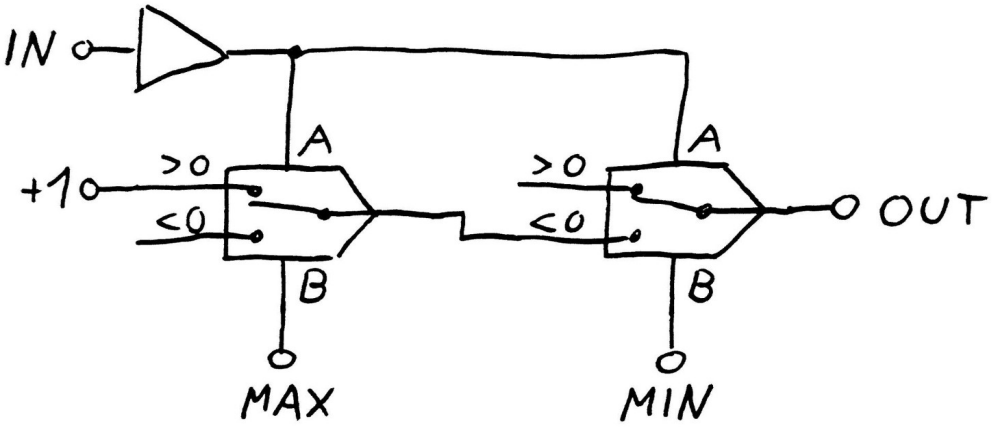
3.7 Four-Quadrant Division

See also: https://analogparadigm.com/downloads/alpaca_39.pdf



3.8 Check if a Signal is between MIN and MAX

Check if IN is between MIN and MAX

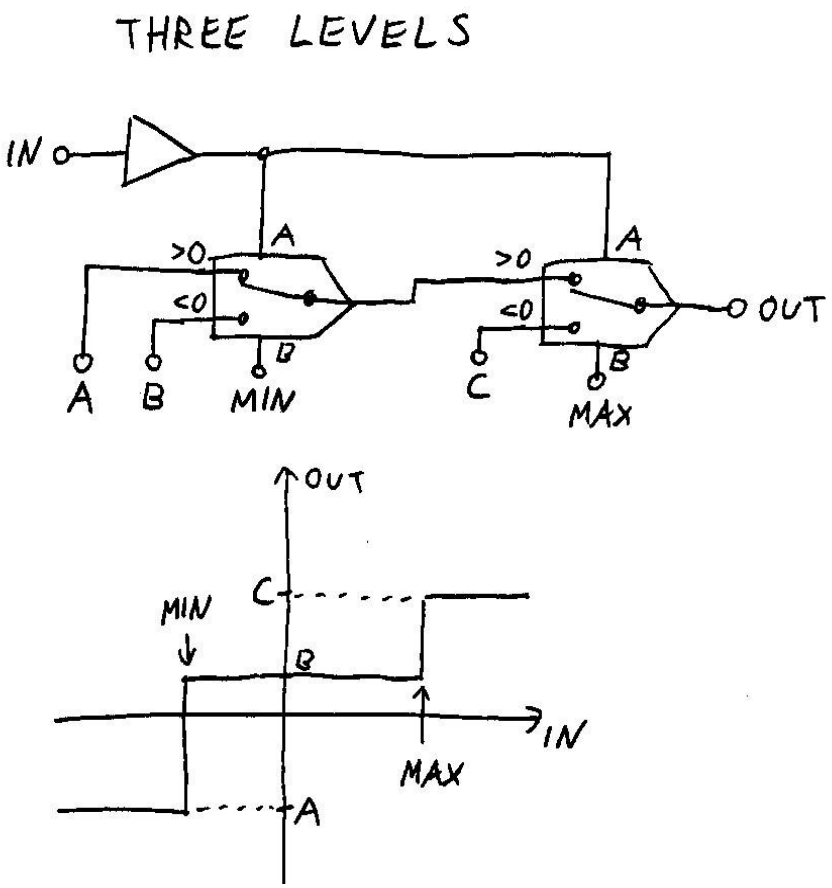


OUT=1, if (IN>MIN) and (IN<MAX)

OUT=0, if (IN>MAX) or (IN<MIN)

3.9 Three Levels

The output is A, B or C, depending if the input signal is below or above the MIN and MAX values. A, B and C can be functions of IN.



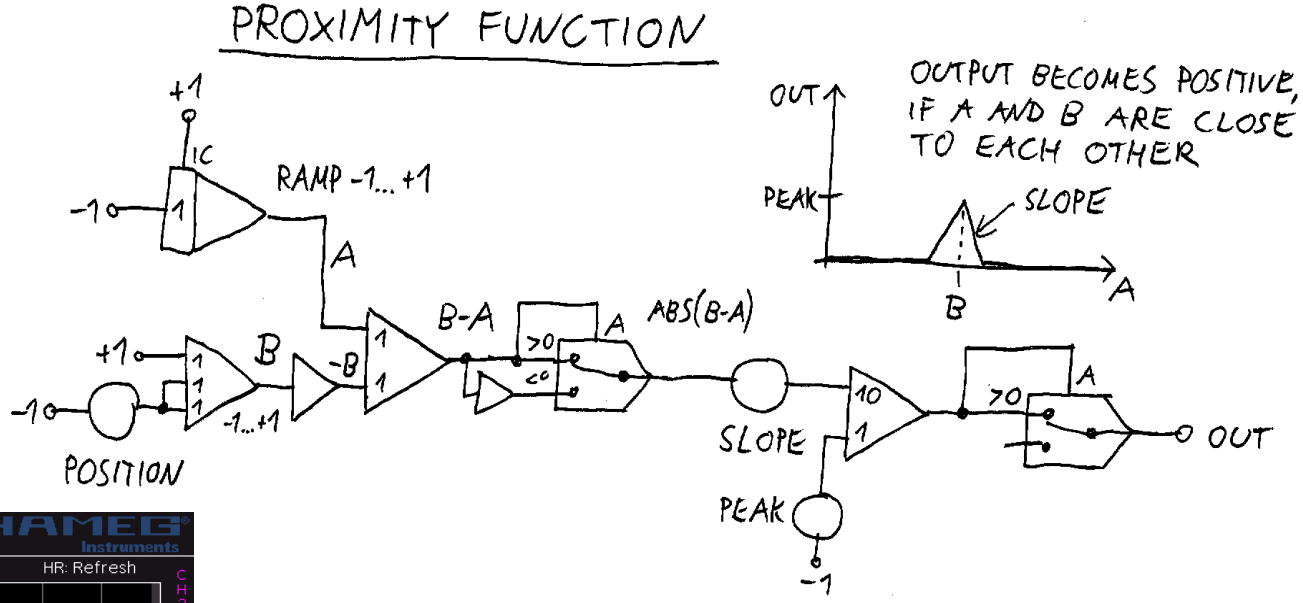
3.10 Proximity Functions

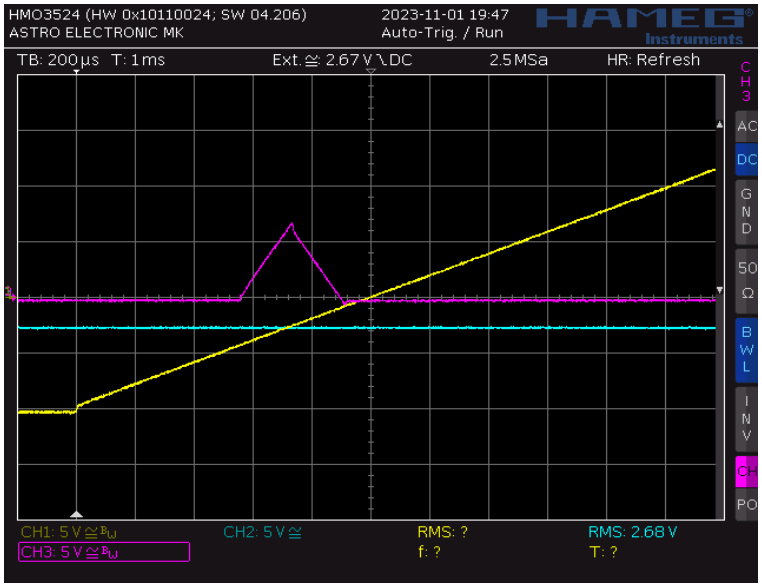
The output becomes positive, if the two inputs A and B are close to each other. Peak level and slope are adjustable.

Yellow: Input A

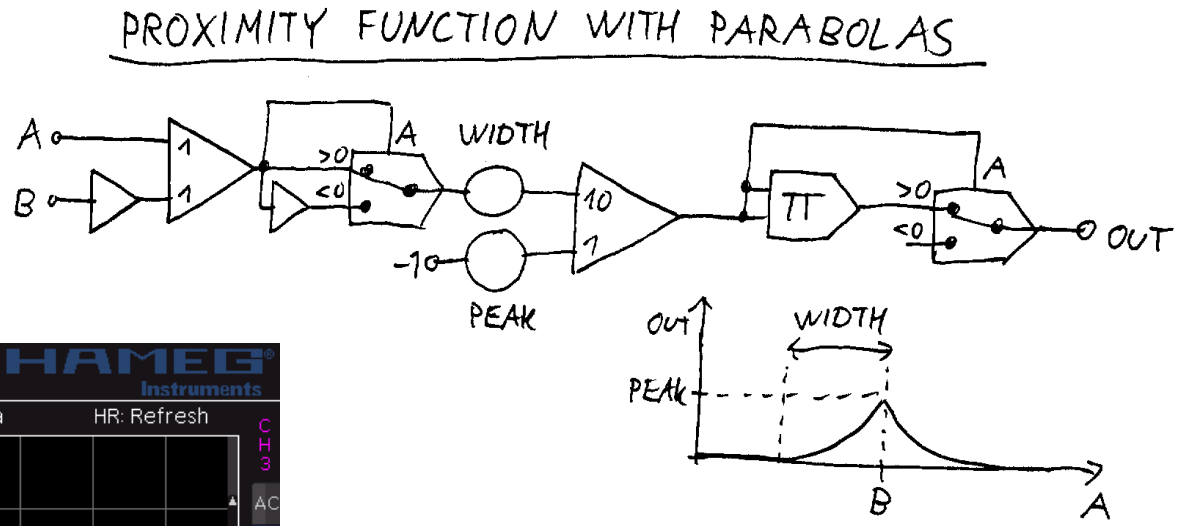
Cyan: Input B

Magenta: Output





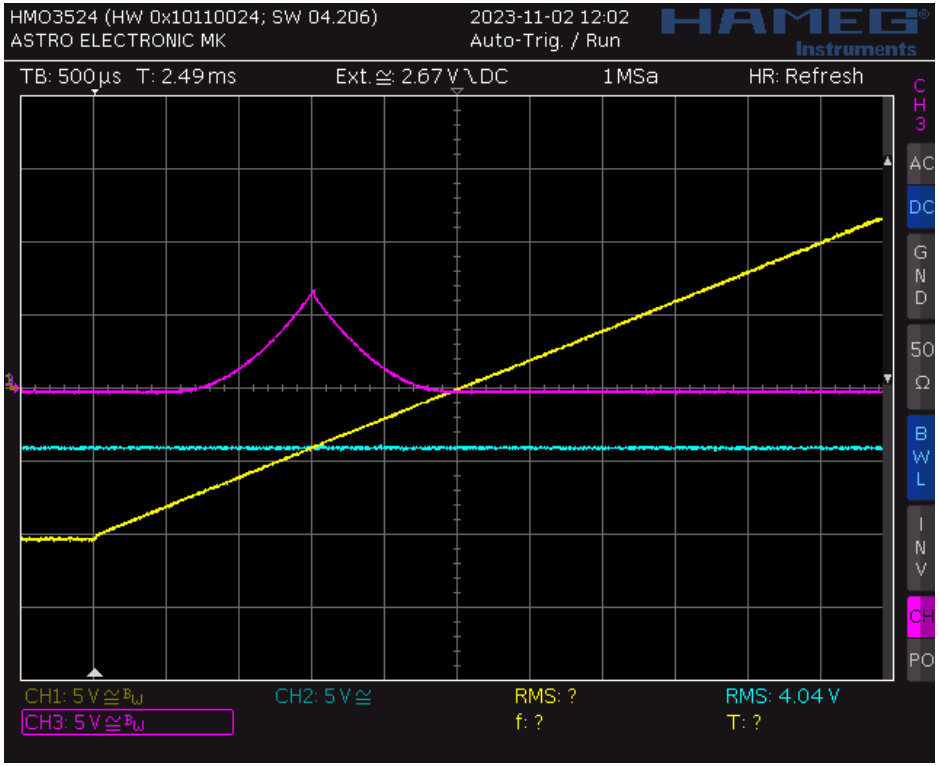
Proximity function with parabolas:



Yellow: Input A

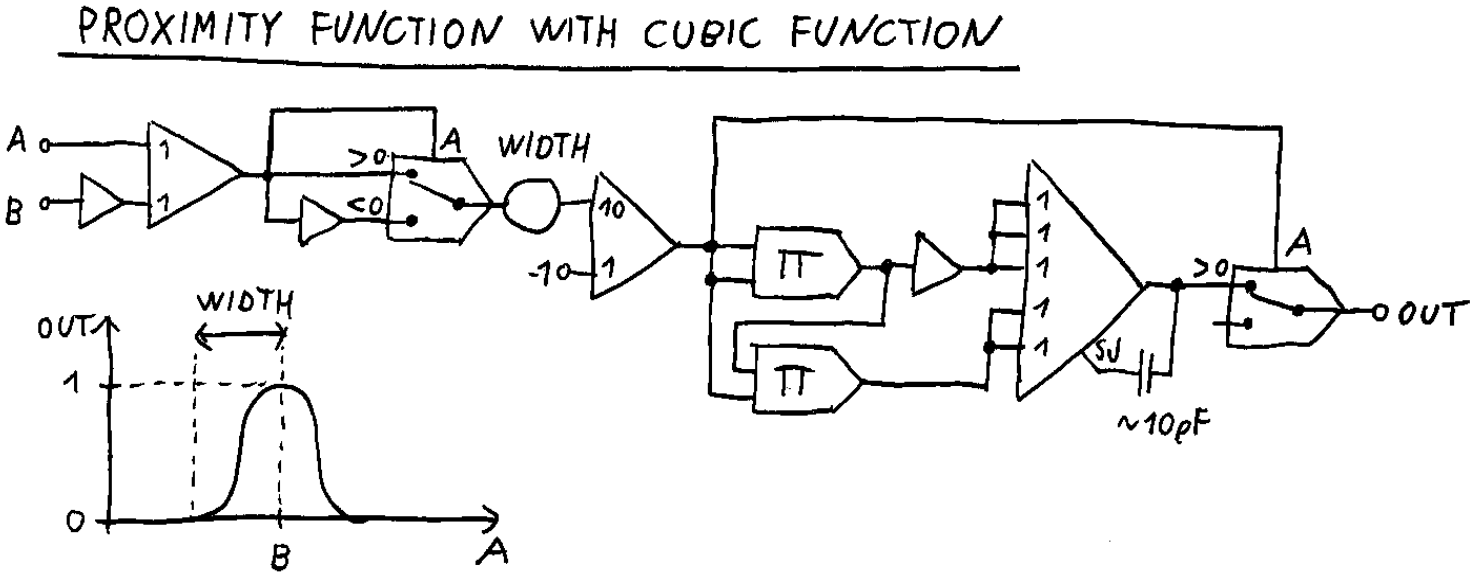
Cyan: Input B

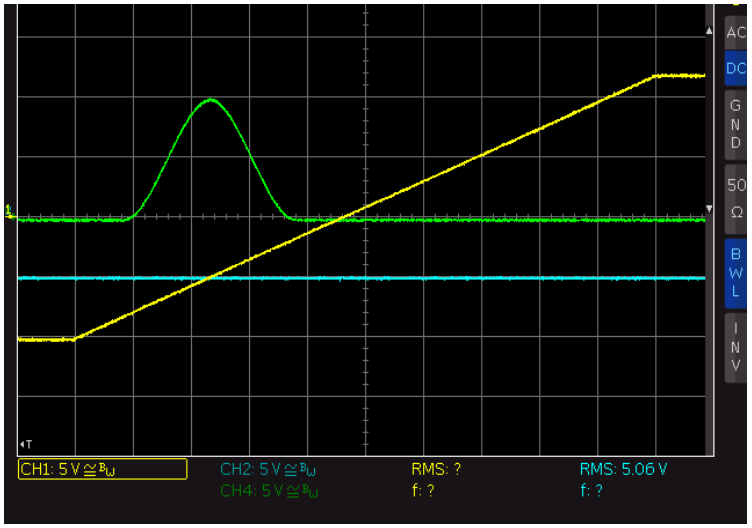
Magenta: Output



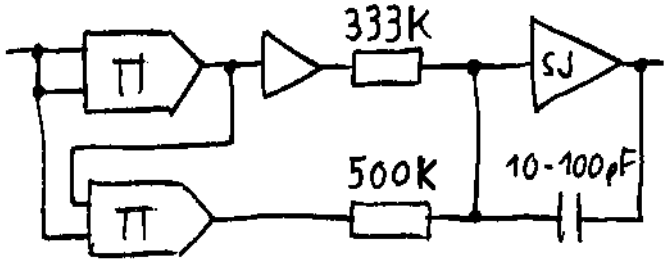
Proximity function with cubic function (approximated Gauss curve):

Note: For small width it's better to increase the feedback capacitor to 100pF.





The summer with 5 inputs can be replaced by an inverter with SJ input, if two additional resistors are used:

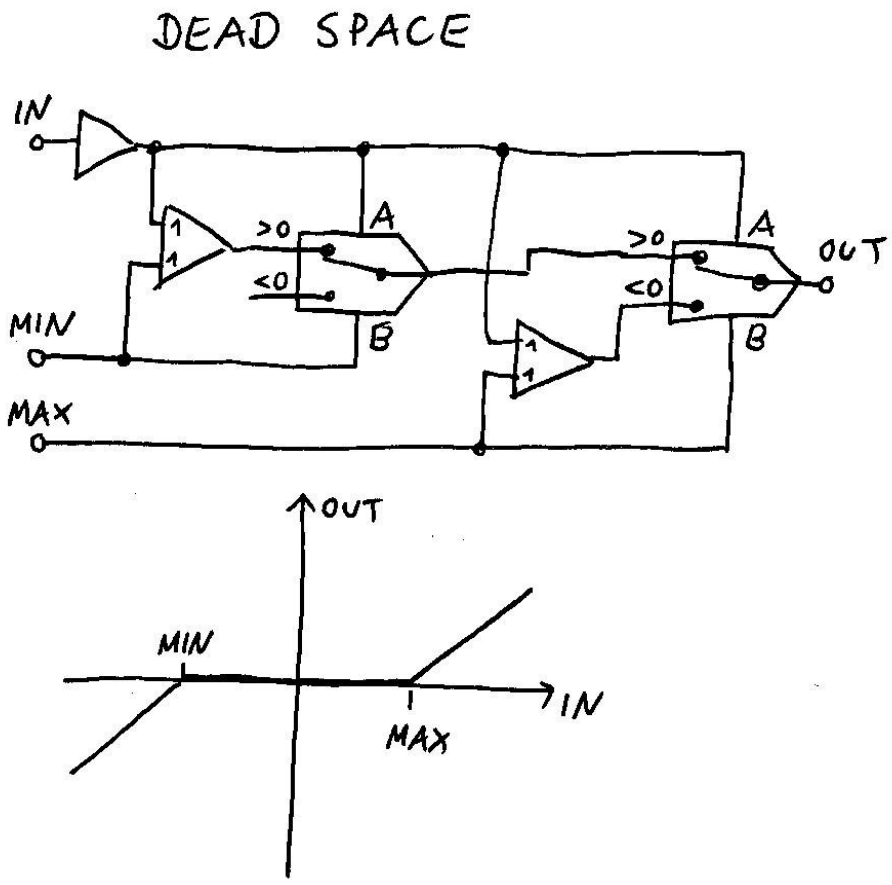


3.11 Dead Space

This is a dead space (dead band?) circuit. The same function can also be realized with two open amplifiers, one summer and 4 diodes, see figure 5.23 in Bernd's book.

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 161ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

See also: See also: Ammon, Werner: Schaltungen der Analogrechentechnik, Page 96: https://www.analogmuseum.org/library/Ammon_Schaltungen.pdf



3.12 Backlash

This works, but who knows how to make the same thing with comparators?

See also: EAI Handbook, page 188 in the PDF:
https://analogmuseum.org/library/eai_handbook.pdf

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 171ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

See also: See also: Ammon, Werner: Schaltungen der Analogrechentechnik, Page 100:
https://www.analogmuseum.org/library/Ammon_Schaltungen.pdf

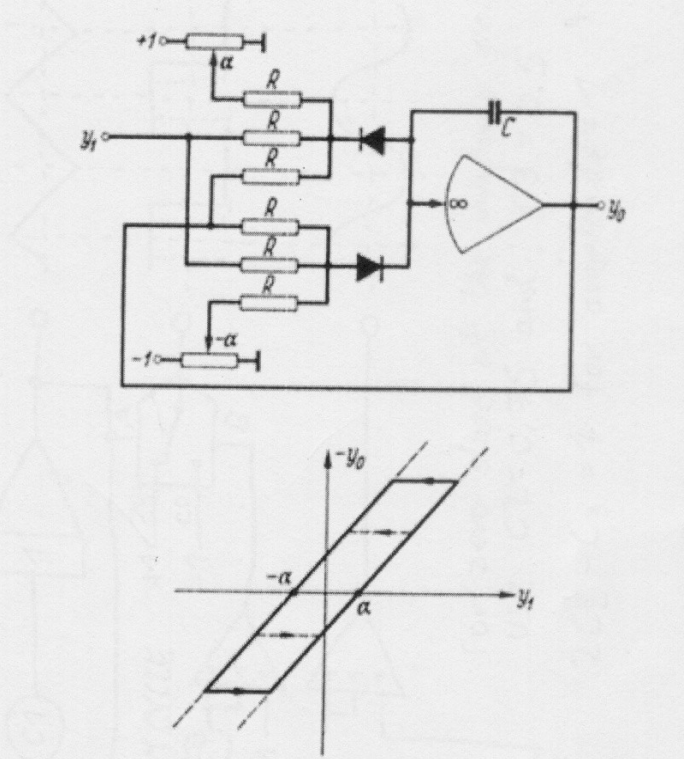


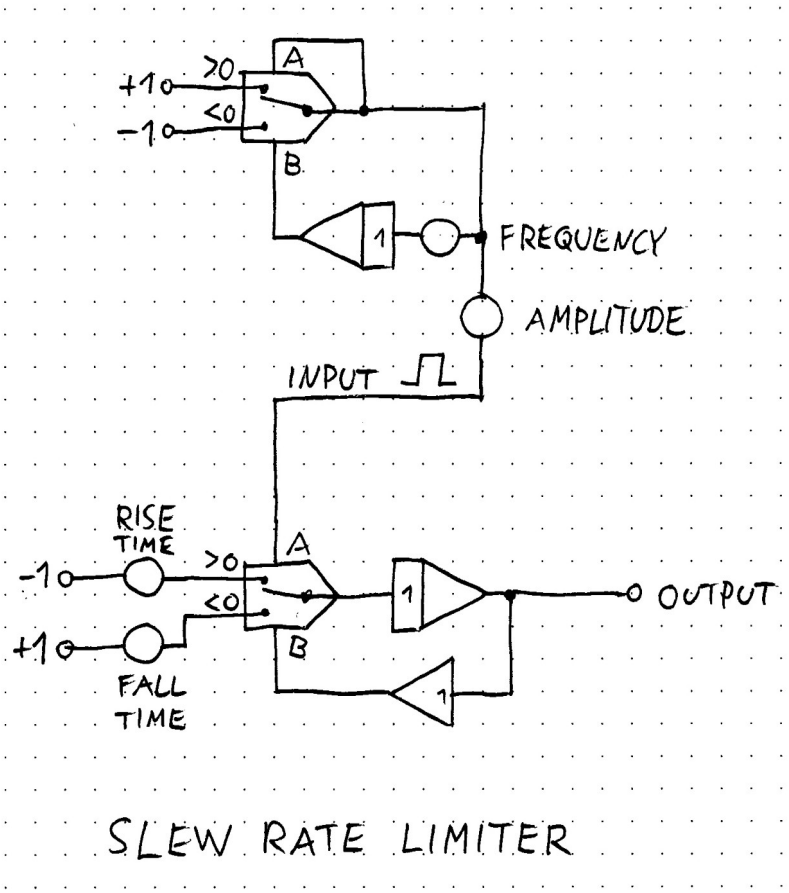
Abb. 3.2-35. Darstellung von Getriebelose (toter Gang).

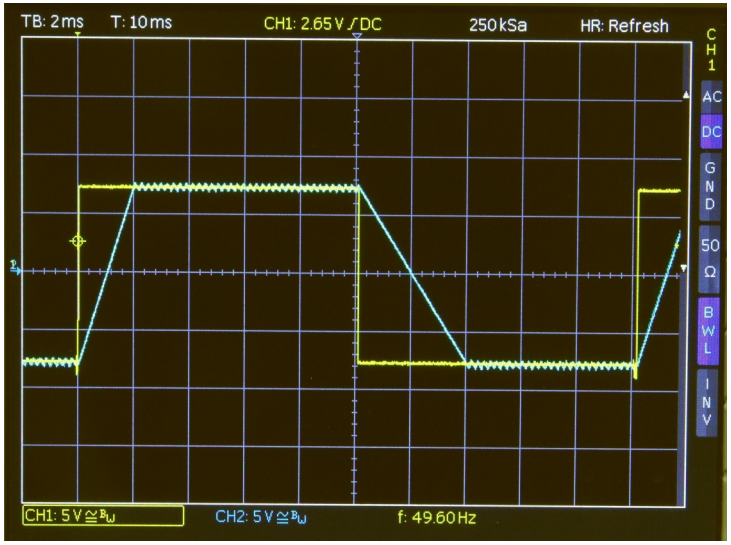
3.13 Coulomb and Static Friction

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 187ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

3.14 Slew Rate Limiter

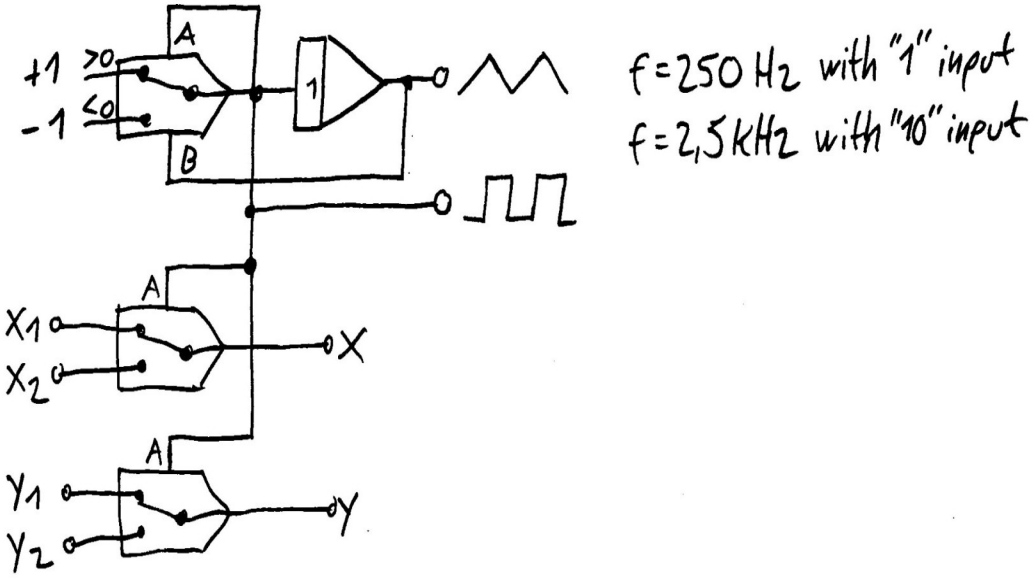
The upper half of this circuit is a simple square wave generator, which makes the yellow signal on the oscilloscope. The lower half is a slew rate limiter with adjustable rise and fall times. Its output is the blue signal on the oscilloscope.



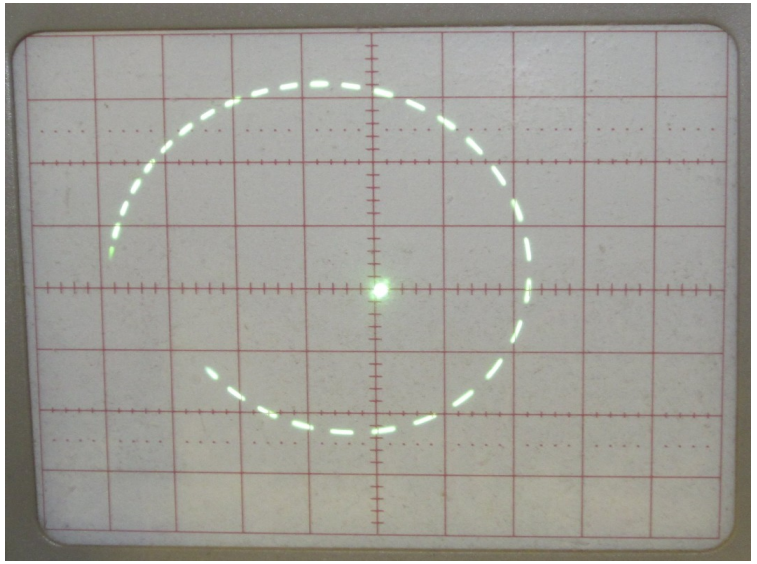


3.15 Multiplexer

This is a two-channel multiplexer for showing two dots on the x/y oscilloscope screen. The switching frequency can be selected by using the 1 or 10 input of the integrator. I did use the multiplexer for the planet orbit example, which I posted some time ago. One dot for the planet and the other for the sun in the center. With the low switching frequency (250Hz), you can see a dotted line, with the planet's speed proportional to the dot length. With the fast switching frequency it becomes a solid line.



f=250 Hz with "1" input
f=2,5 kHz with "10" input



3.16 Low Pass Filter

Integration Resistor	Integration Capacitor	Integration time constant TC
100 kΩ ("10" Input)	1nF (normal)	100 µs
1 MΩ ("1" Input)	1nF (normal)	1 ms
1 MΩ ("1" Input)	10nF (external)	10 ms
1 MΩ ("1" Input)	100nF (SLOW)	100 ms
10 MΩ (external)	100nF (SLOW)	1 s
100 MΩ (external)	100nF (SLOW)	10 s
1 GΩ (external)	100nF (SLOW)	100 s
1 GΩ (external)	1µF (with open amplifier)	1000 s
1 GΩ (external)	10µF (with open amplifier)	10000 s

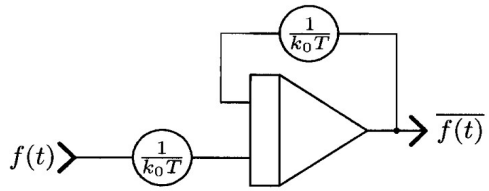


Fig. 5.11. Basic circuit of a low pass filter

Source of the image: Bernd Ulmann: Analog and Hybrid Computer Programming, page 83

f_{FILTER} = K / (2π · k_0) where K are the two coefficients (at the input and in the feedback loop) and k_0 is the time constant of the integrator.

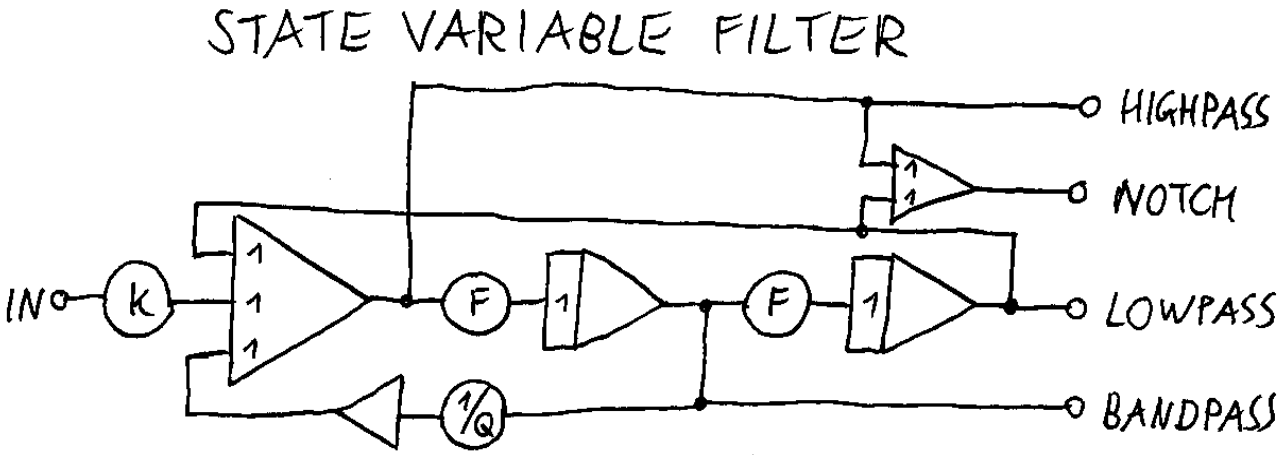
Please note that this isn't the correct definition of k_0, as can be found in the book "Analogrechnen" by Giloi and Lauber (on the page before page 1, and on page 65): k_0 = 1 / (R · C), which means the unit of k_0 is [1/s], so that k_0 equals the inverse of the integrator's time constant.

U_{OUT} / U_{IN} = 1 / sqrt(1 + (f_{IN} / f_{FILTER})^2)

f_{IN} / f_{FILTER}	0.1	0.25	0.5	1.0	2.0
U_{OUT} / U_{IN}	0.995	0.970	0.894	0.707	0.447

It's better to use this formula instead: f_{FILTER} = K / (2π · TC) where TC is the time constant of the integrator.

3.17 State Variable Filter (Lowpass, Highpass, Bandpass, Notch)



Center frequency $f_0 = F / (2\pi \cdot TC)$, where TC is the time constant of the integrators.

The bandwidth is $BW = f_2 - f_1$. The center frequency is $f_0 = \text{sqrt}(f_1 \cdot f_2)$. The quality factor is $Q = f_0 / BW$.

For lowpass, highpass and notch filter the gain is equal K.

For bandpass filter the gain at the center frequency is equal K·Q. That means K should be smaller or equal 1/Q, so that the output can't overflow when the input uses the full machine unit range.

A notch filter can be realized by summing the lowpass and highpass outputs of a state variable filter. To compensate for the small gain (if K is small), you can use the "10" inputs of the summer for the notch signal.

If you want to control the filter's center frequency by a voltage, you can simply replace both "F" coefficients by multipliers. But keep in mind that the bandwidth is proportional to the center frequency. So it's not a good idea to build a spectrum analyzer with this filter (unless the frequency span is very small).

See also chapter 11.7 in this book: https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Operational_Amplifiers_and_Linear_Integrated_Circuits_-_Theory_and_Application_%28Fiore%29/11%3A_Active_Filters/11.07%3A_Band-Pass_Filter_Realizations

Formulas for the bandpass filter:

Gain: $U_{OUT} / U_{IN} = (K \cdot \Omega) / \text{sqrt}(1 + \Omega^2 \cdot (1 / Q^2 - 2) + \Omega^4)$

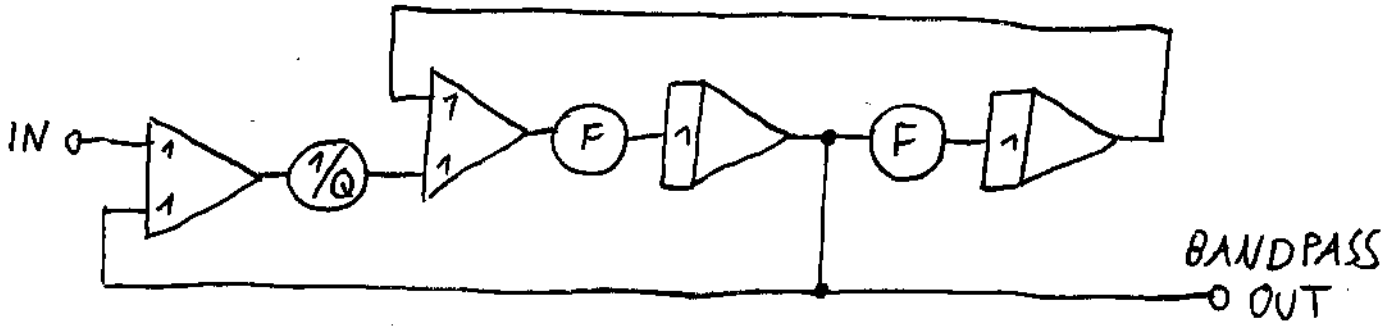
Phase shift: $\varphi = \text{atan}(Q \cdot (1 - \Omega^2) / \Omega)$

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 9. Auflage, page 426

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 872

The drawback of the previous circuit is that the 1/Q coeffiecient does not only change the quality factor, but also the gain at the center frequency. This can be compensated by simultaneously changing also the K coefficient.

In the following circuit the quality factor can be adjusted with one coefficient. The gain at center frequency is always 1.



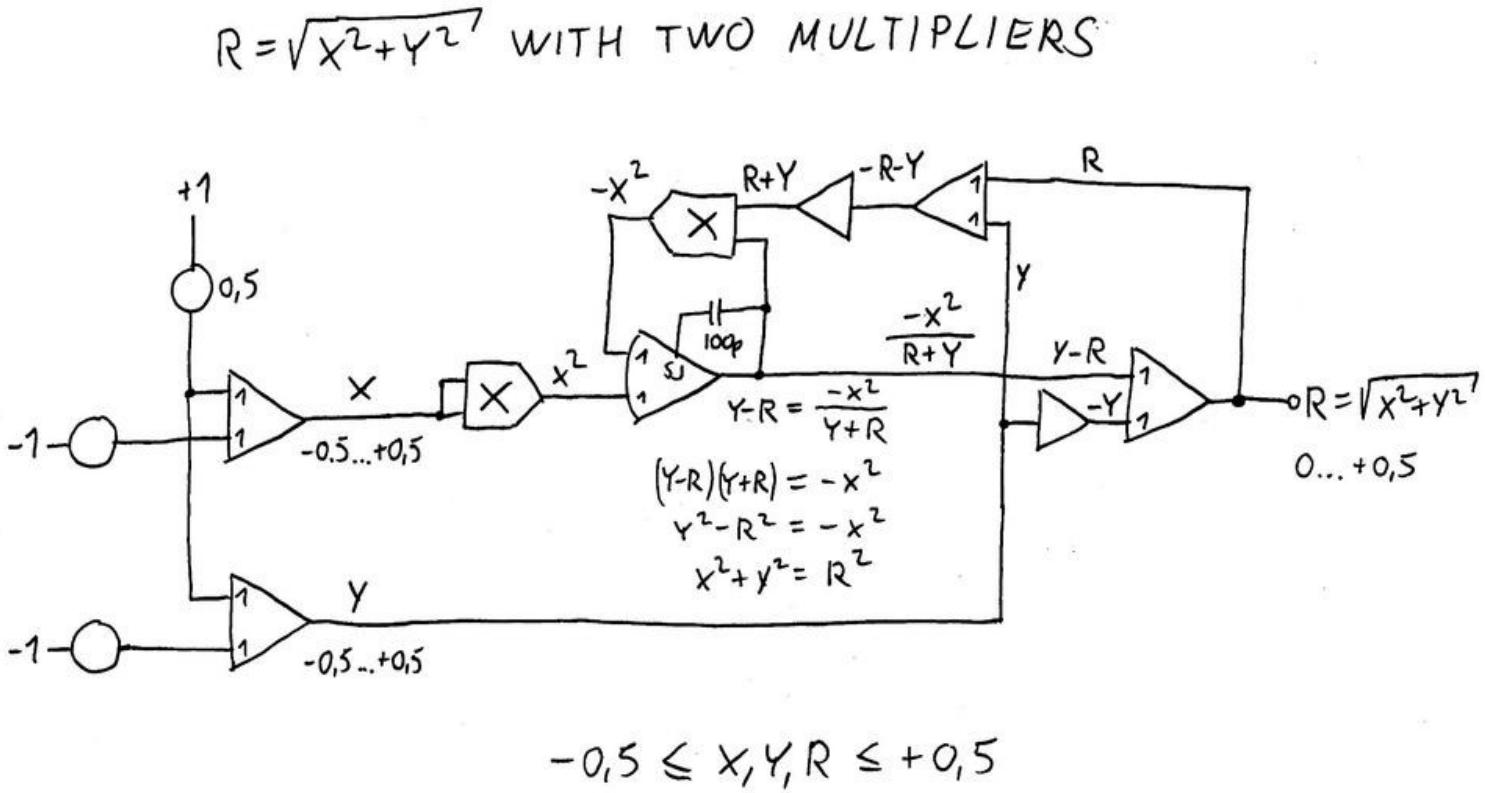
3.18 Allpass Filter

See also: https://www.analog.com/media/en/training-seminars/tutorials/MT-202.pdf

3.19 Vector Length with two Multipliers

Calculate the length of a two-dimensional vector with two multipliers. It's important that the X and Y signals must not exceed the -0.5 to +0.5 range. Otherwise the -R-Y and R+Y signals would overflow.

I found the example here: http://www.analogmuseum.org/library/pythagoras.pdf

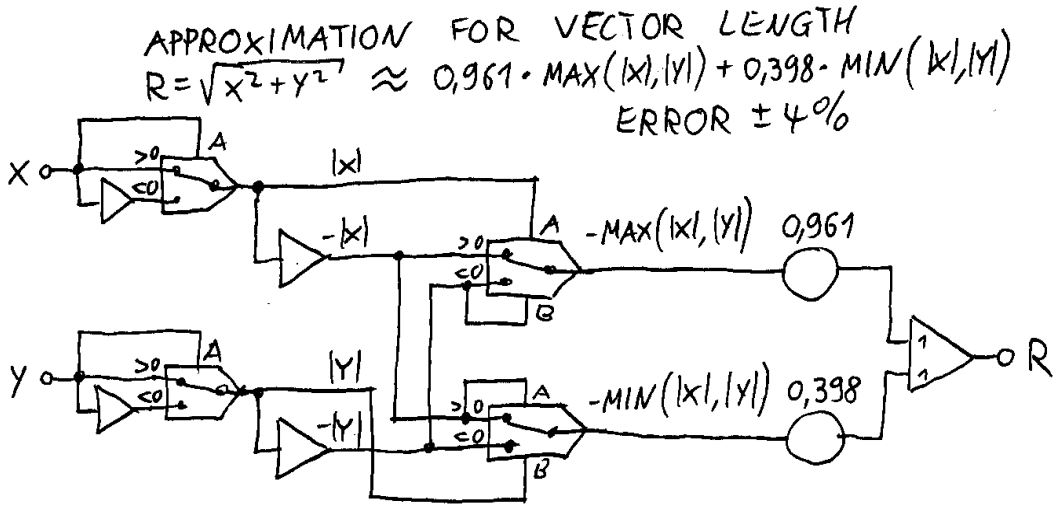


See also: https://www.analog.com/media/en/analog-dialogue/volume-6/number-3/articles/volume6-number3.pdf#page=3

See also: https://rclab.de/analogrechner/vektorlaenge

3.20 Approximations for Vector Length

Approximation for $r = \sqrt{x^2 + y^2}$	Error range for a point on the unit circle r = 1	Error in %	Required components in THAT
$r \approx \max(\lvert x\rvert, \lvert y\rvert)$	0.707...1.000 largest error at 45°	-29.3%...+0%	3 Comparators 3 Inverters
$r \approx \max(\lvert x\rvert, \lvert y\rvert) + 0.414 \cdot \min(\lvert x\rvert, \lvert y\rvert)$ Note: 0.414 = sqrt(2) - 1	1.000...1.082 largest error at 22.5°	-0.0%...+8.2%	4 Comparators 4 Inverters 1 Coefficient 1 Summer
$r \approx \max(\lvert x\rvert, \lvert y\rvert) + 13/32 \cdot \min(\lvert x\rvert, \lvert y\rvert)$	0.994...1.079 largest error at 22.1°	-0.6%...+7.9%	same as above, but optimized for 8-bit 8051 microcontroller
$r \approx 0.960 \cdot \max(\lvert x\rvert, \lvert y\rvert) + 0.398 \cdot \min(\lvert x\rvert, \lvert y\rvert)$	0.960...1.039 balanced positive and negative errors at 0°, 22.5° and 45°	-4.0%...+3.9%	4 Comparators 4 Inverters 2 Coefficients 1 Summer



See also: Bronstein-Semendjajew: Taschenbuch der Mathematik, 21. Auflage, Page 101

3.21 Function Generators

In analog computing the term "Function Generator" means a function y = f(x) where x is an input signal and not y = f(t) where t is the time.

See chapter 2.6 in Bernd Ulmann: Analog and Hybrid Computer Programming

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 9. Auflage, page 345ff

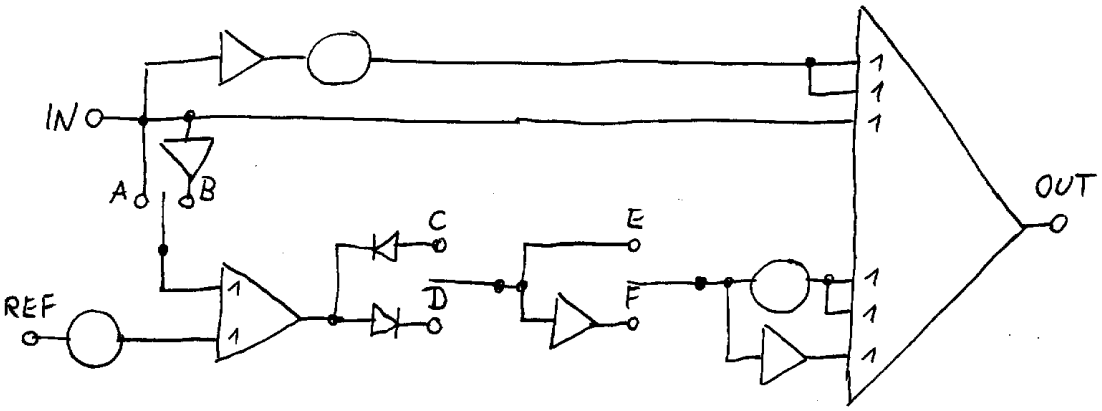
See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 797ff

User-defined transfer functions can easily be realized with hybrid analog / digital circuits, see my example with Teensy LC.

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 165ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

In this simplified circuit the upper coefficient sets the default slope, if no bend pointr is active. The lower part is for a bend point and can be repeated many times for more bend points.

Input	Ref	Diode	Output	Position of bend point	Is slope left or right of bend point?	Slope, if coefficient = 1
A	+1	C	E	neg	right	pos
A	+1	C	F	neg	right	neg
A	+1	C	E	neg	left	pos
A	+1	C	F	neg	left	neg
A	+1	D	E	pos	right	pos
A	+1	D	F	pos	right	neg
A	+1	D	E	pos	left	pos
A	+1	D	F	pos	left	neg
B	-1	C	E	pos	left	neg
B	-1	C	F	pos	left	pos
B	-1	C	E	pos	right	neg
B	-1	C	F	pos	right	pos
B	-1	D	E	neg	left	neg
B	-1	D	F	neg	left	pos
B	-1	D	E	neg	right	neg
B	-1	D	F	neg	right	pos



How to make a bend point switchable:

-- Positive position, the slope is right of the bend point, positive slope if coefficient = 1

-- Negative position, the slope is left of the bend point, positive slope if coefficient = 1

Three things must be switched: Input polarity A/B, Reference +1/-1, and output polarity E/F.

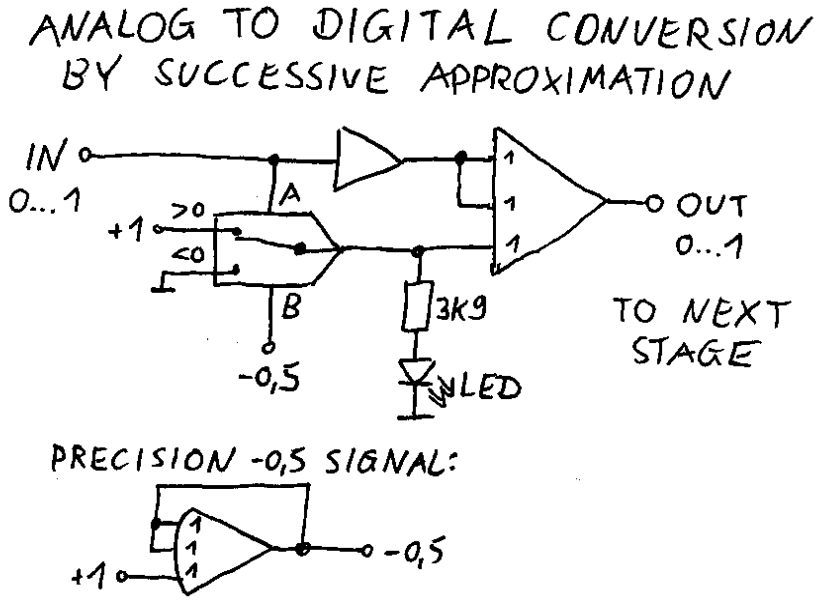
3.22 Analog to Digital Conversion by Successive Approximation

This circuit can be cascaded. Because each bit requires a comparator, only two bits are possible with one THAT. With two THATs you can realize a 4-bit converter, and so on.

Each stage works as follows:

- **If the input is less than 0.5, the digital bit is 0 and the output is the input multiplied by two.**
- **If the input is greater or equal 0.5, the digital bit is 1 and the output is the input multiplied by two, minus one.**

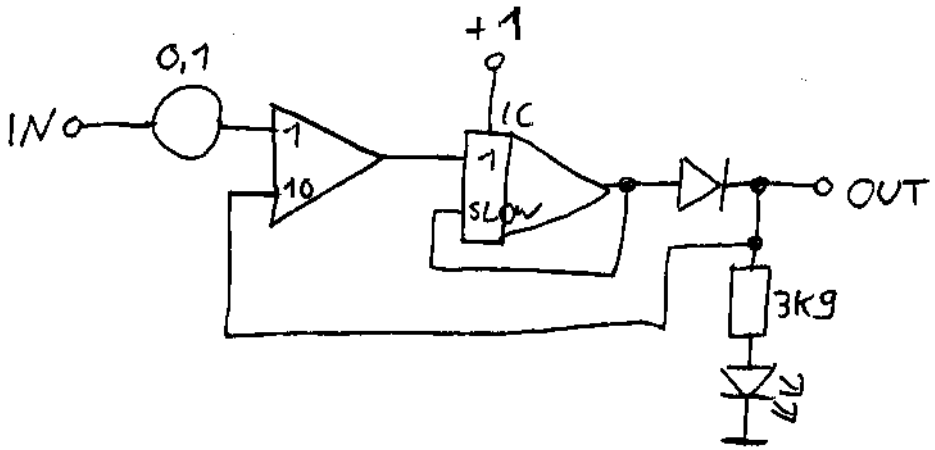
The output of the last stage can be divided by a suitable power of two and then becomes the remainder.



The advantage of the precision -0.5 signal is that it doesn't need an adjustment with a coefficient.

3.23 Switch on Lights, one after the other

Connect the output to the input of the next stage. The first stage gets +1 as input.

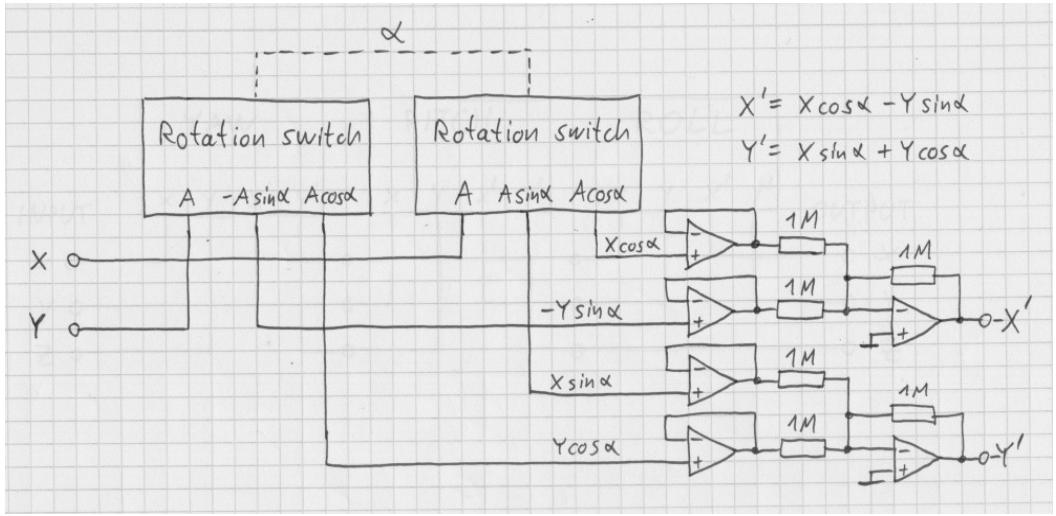


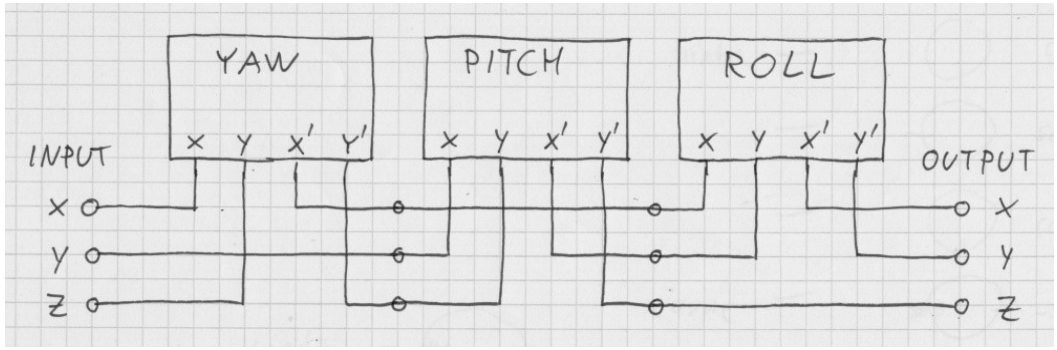
3.24 Voltage to Time Conversion

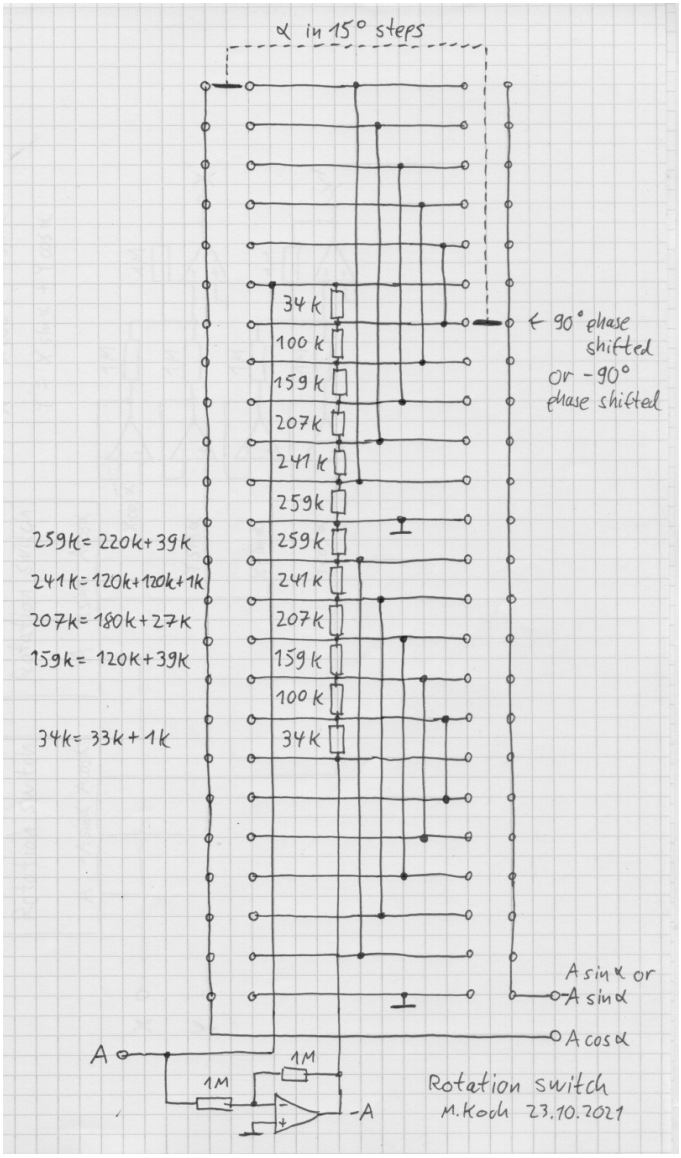
- **If a variable control voltage is connected to the input of an integrator and we stop the integration when a fixed output voltage U_{max} is reached, the required time is inversely proportional to the input voltage. With other words: Voltage is proportional to frequency.**
- **If a constant is connected to the input of an integrator and we stop the integration when the variable control voltage is reached, the required time is proportional to the input voltage.**

3.25 Rotation Matrix in 3D Space

4x24 rotary switches are available from Aliexpress:
https://de.aliexpress.com/item/1005001651673638.html





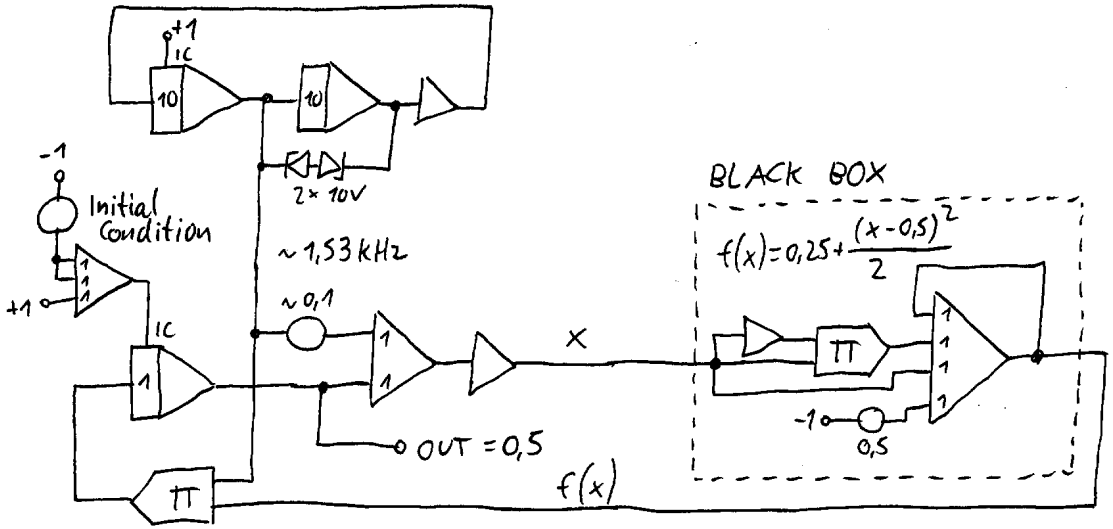


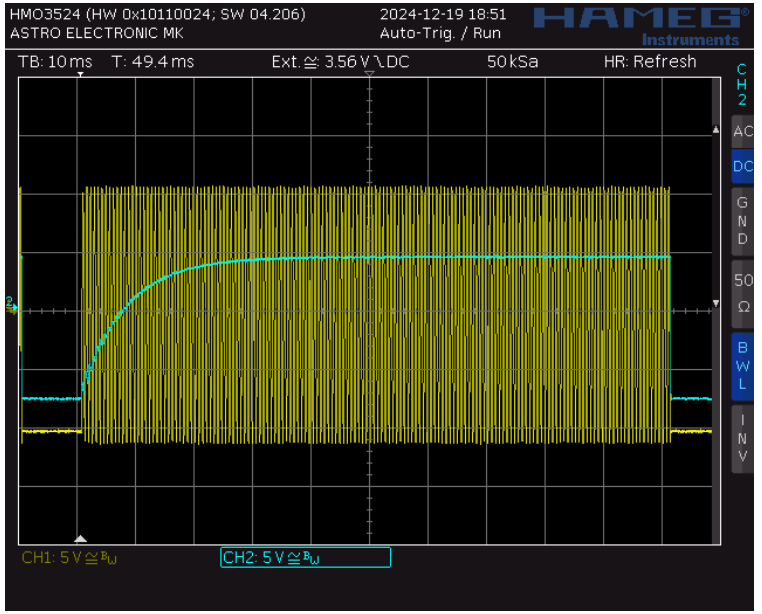
3.26 Optimization Problems

Each optimization problem is equivalent to a minimum-finding problem.

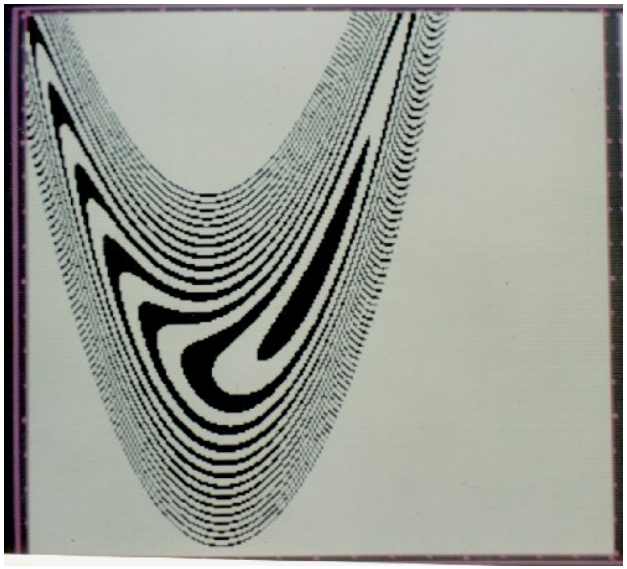
See also: R. J. McGrath and V. C. Rideout: A Simulator Study of a Two-Parameter Adaptive System
https://ieeexplore.ieee.org/abstract/document/6429307

Here is a working example for an optimization problem. The function in the black box has a minimum at x = 0.5. The circuit finds the minimum regardless which initial condition is used. A small sine wave signal is added to the input value x, and the output f(x) is multiplied by the reference sine wave. The product is used to correct the x value up or down, until the minimum is found. In the above paper is written that this approach can also be used to optimize several parameters simultaneously, if different frequencies are used.





3.26.1 Two-dimensional Test Function



We used this function in 1988 at Braunschweig university as a test function for various minimum finding algorithms:

$f(x,y) = 5 \cdot x^4 - 10 \cdot x^2 \cdot y + x^2 + 5 \cdot y^2 - 2 \cdot x + 5$

The minimum is at x = 1, y = 1. Good initial conditions for testing are (x = -1, y = 2.5) or (x = 1.5, y = 3).

The picture shows the topography of this function in the range x = [-2 ... +4] and y = [-2 ... +4], with the minimum in the center at (1, 1)

Let's scale the variables x and y to machine variables: $x_m = x/2.5$, $y_m = y/2.5$, $x = 2.5 \cdot x_m$, $y = 2.5 \cdot y_m$

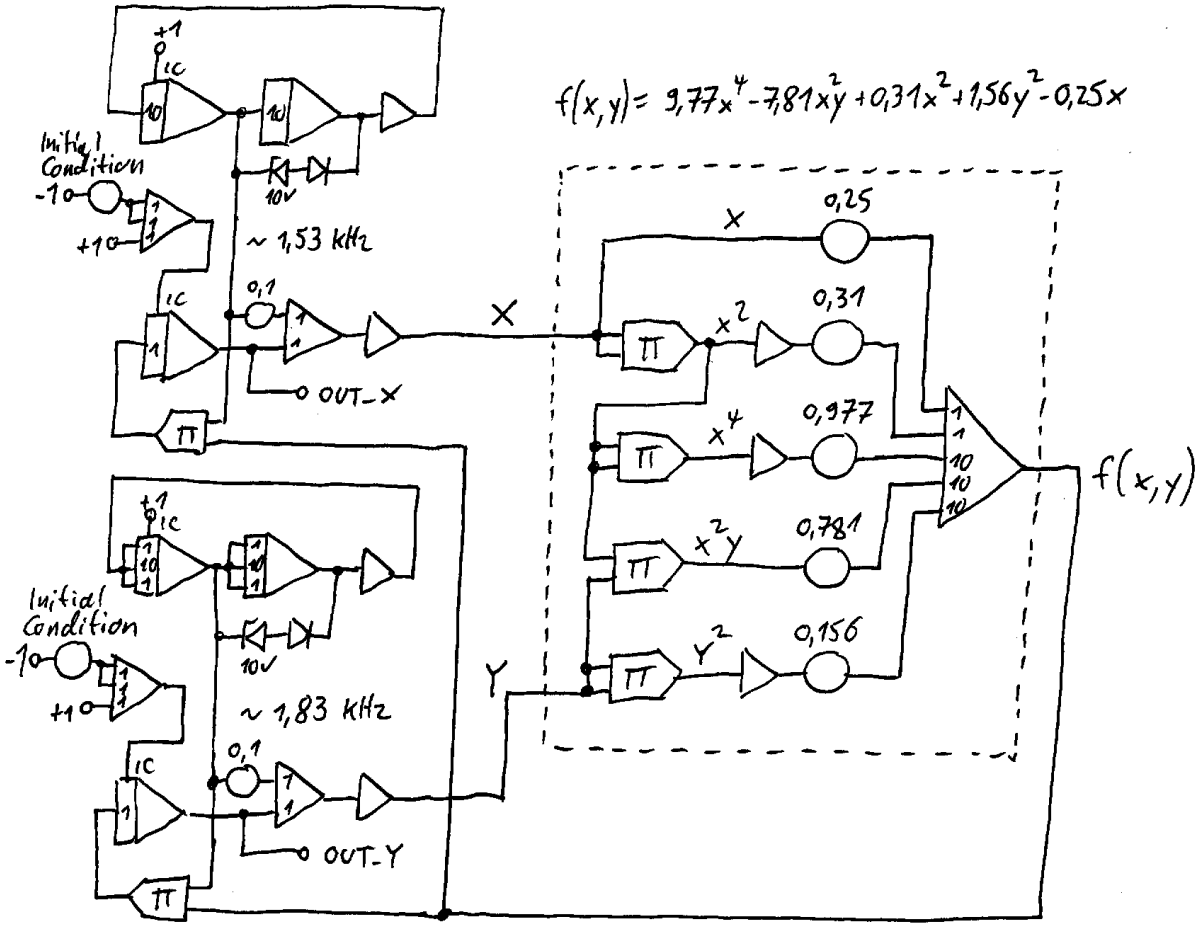
$f(x_m,y_m) = 195.31 \cdot x_m^4 - 156.25 \cdot x_m^2 \cdot y_m + 6.25 \cdot x_m^2 + 31.25 \cdot y_m^2 - 5 \cdot x_m + 5$

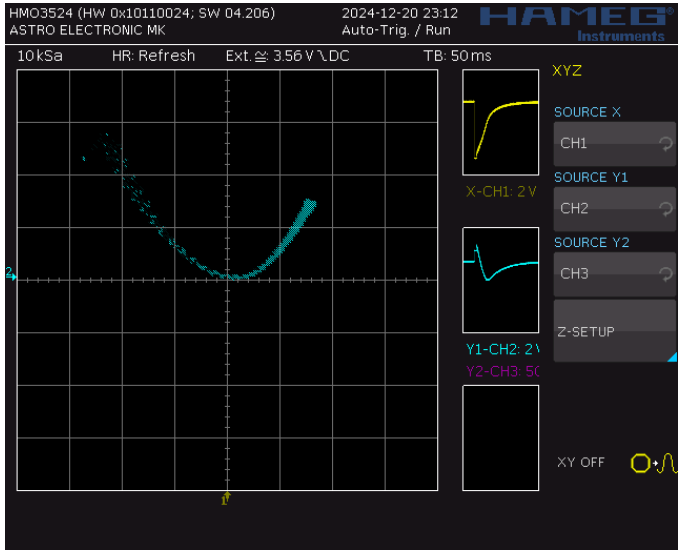
Divide by 20 to make the function's value smaller (without changing the minimum's position), and remove the unnecessary constant term:

$f(x_m,y_m) = 9.766 \cdot x_m^4 - 7.812 \cdot x_m^2 \cdot y_m + 0.312 \cdot x_m^2 + 1.562 \cdot y_m^2 - 0.25 \cdot x_m$

The minimum is at $x_m = 0.4$, $y_m = 0.4$. Good initial conditions for testing are ($x_m = -0.4$, $y_m = 1$) or ($x_m = 0.6$, $y_m = 1$).

Two-dimensional minimum finding:

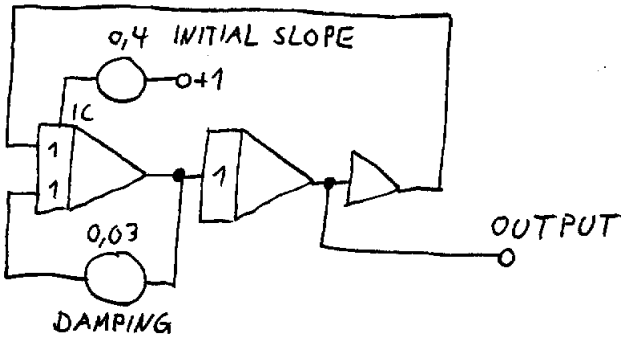




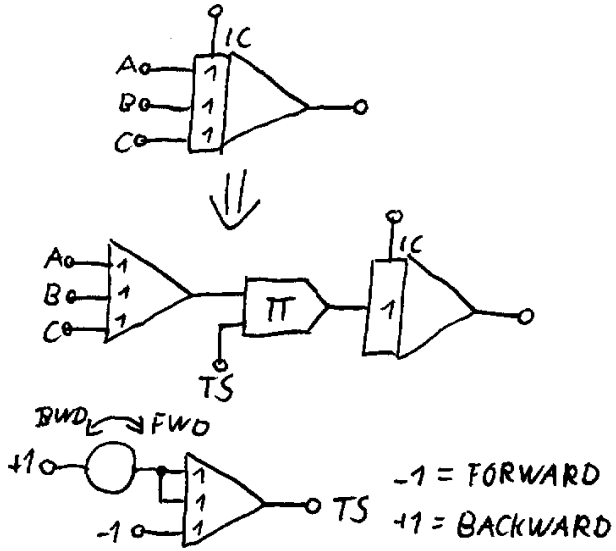
3.27 Backward in time simulation

Any simulation circuit can be changed from forward-in-time to backward-in-time simply by inverting all inputs of all integrators (but the initial conditions remain unchanged). This is the same as inverting the integrator's output and inverting the initial condition.

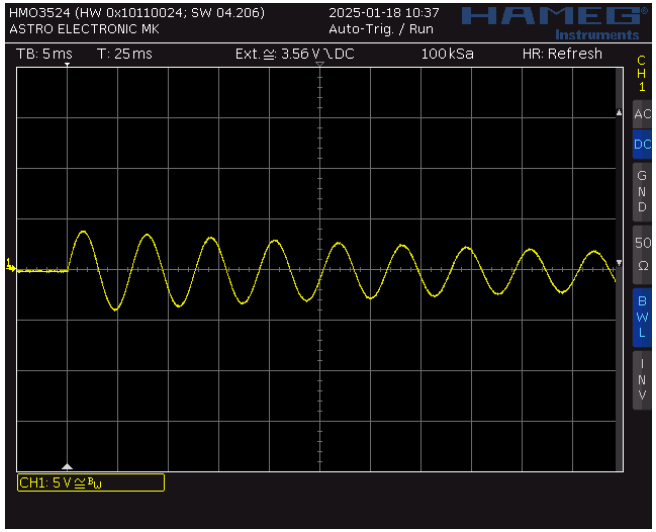
Simple example for damped oscillator, forward in time:



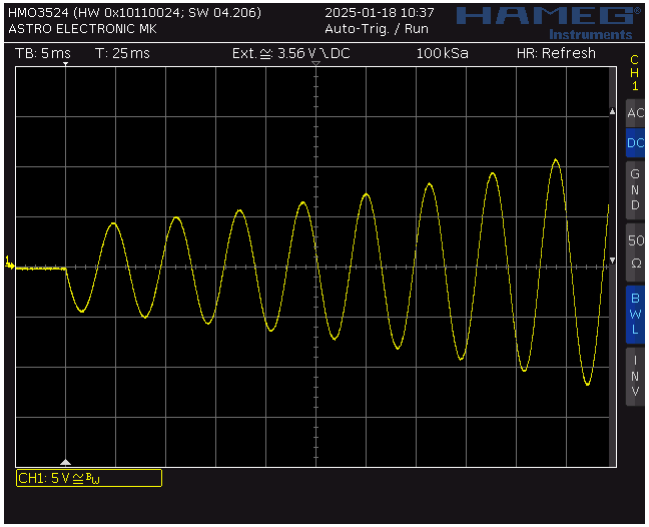
Replace each integrator by the circuit shown below:



Forward in time:



Backward in time:



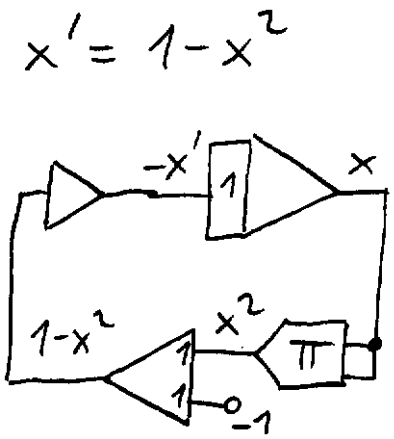
Many circuits run into overflow either forward-in-time or backward-in-time. It's a pity that all circuits for delay time approximation seem to be instable when used backward-in-time. I wanted to feed a bitstream of tomorrow's lottery numbers to the input of the delay circuit, and then run a backward-in-time simulation. Unfortunately it seems to be impossible.

But there are also circuits which converge to a stable solution in both directions of time.

Comment from Matthias Weiss:

"Look for a dynamical system with two fixed points, one stable and one unstable. If you successfully reverse the time, the same initial condition should take you to one or the other fixed point. For example, x' = 1 - x² would have two fixed points at ±1 (where x'(t) = 0). Linearizing around each fixed point gives you 2 at x = -1 (unstable) and -2 at x = 1 (stable)."

This is the circuit:

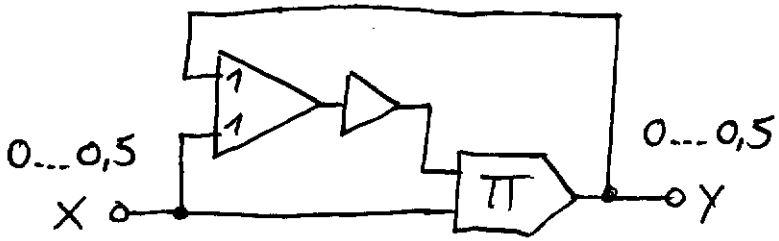


In forward-in-time direction it converges to +1 for all initial conditions.
In backwards-in-time direction (just remove the inverter) it converges to -1 for all initial conditions.

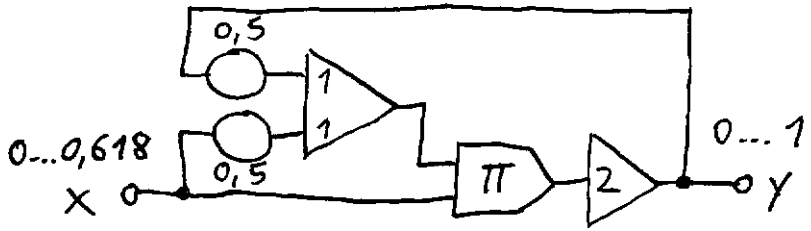
3.28 Nonlinear Functions / Infinite Power Series

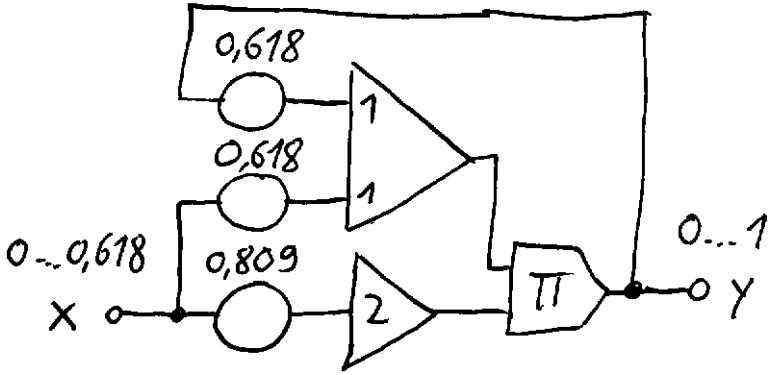
See page 47ff in: Analog Devices: Nonlinear Circuits Handbook, Second Edition January 1976, ISBN: 0-916550-01-X

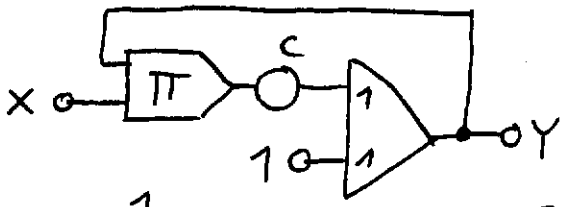
This circuit is for an infinite power series $y = x^2 + x^3 + x^4 + x^5 + ...$



$$y = \frac{x^2}{1-x} = x^2 + x^3 + x^4 + x^5 + \ldots$$



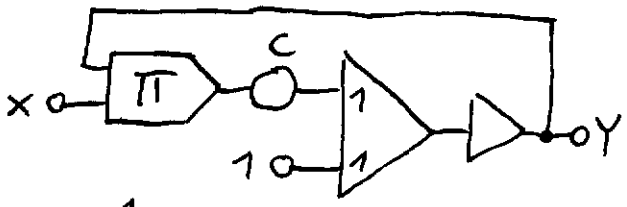




$c=1$

x	Y
0	-1
0,5	-0,666
1	-0,5

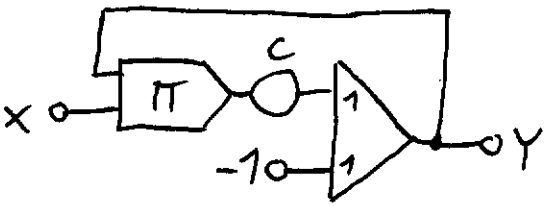
$$Y=\frac{-1}{1+cx}=-1+cx-c^2x^2+c^3x^3-\dots$$



$c=1$

x	Y
-1	0,5
-0,5	0,666
0	1

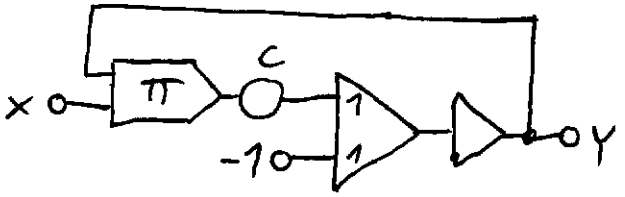
$$Y=\frac{1}{1-cx}=1+cx+c^2x^2+c^3x^3+\dots$$



$c=1$

x	Y
0	1
0,5	0,666
1	0,5

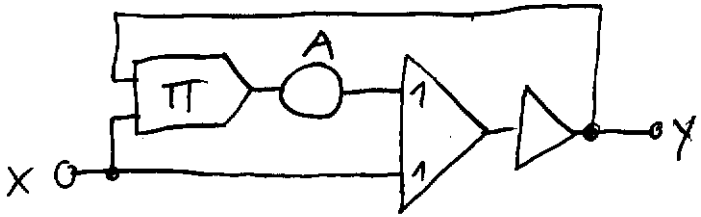
$$Y=\frac{1}{1+cx}=1-cx+c^2x^2-c^3x^3+\dots$$



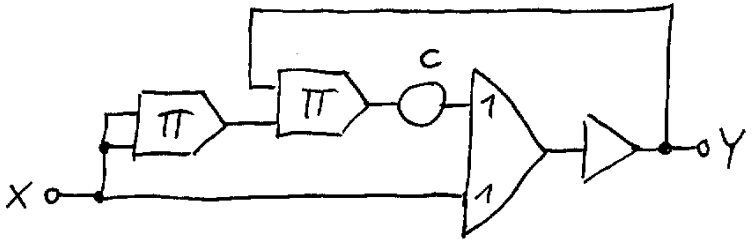
$c=1$

x	Y
-1	-0,5
-0,5	-0,666
0	-1

$$Y=\frac{-1}{1-cx}=-1-cx-c^2x^2-c^3x^3-\dots$$

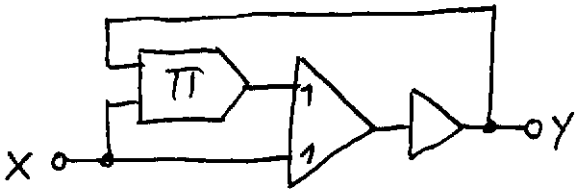


$$y=\frac{x}{1-Ax}=x+Ax^2+A^2x^3+A^3x^4+\dots$$



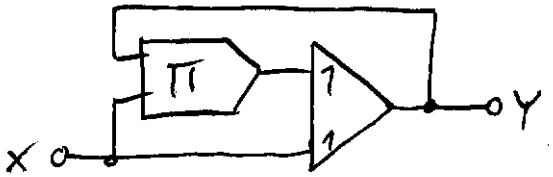
x	Y
-0,618	-1
-0,5	-0,666
-0,25	-0,266
0	0
0,25	0,266
0,5	0,666
0,618	1

$$Y=\frac{x}{1-cx^2}=x+cx^3+c^2x^5+c^3x^7+\dots$$



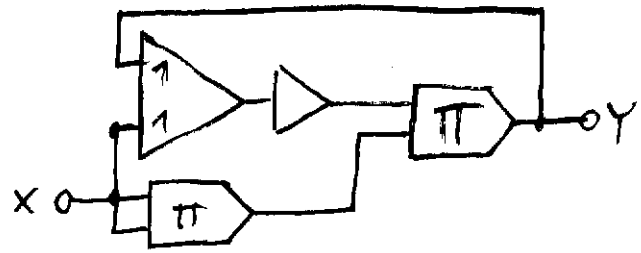
$$Y = \frac{x}{1-x} = x + x^2 + x^3 + x^4 + \dots$$

X	Y
-1	-0,5
-0,75	-0,429
-0,5	-0,333
-0,25	-0,2
0	0
0,25	0,333
0,5	1

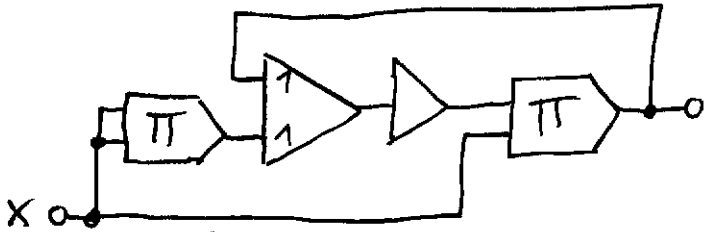


$$Y = \frac{-x}{1+x} = -x + x^2 - x^3 + x^4 - \dots$$

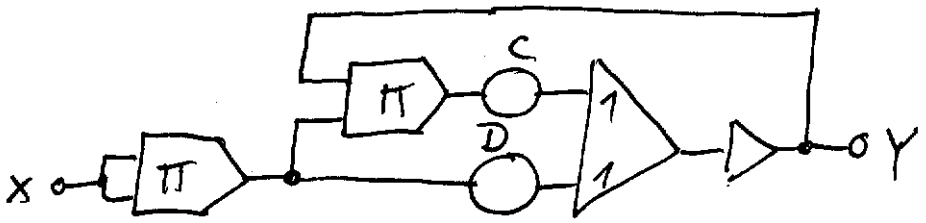
X	Y
-0,5	1
-0,25	0,333
0	0
0,25	-0,2
0,5	-0,333
0,75	-0,429
1	-0,5



$$Y = \frac{x^3}{1-x^2} = x^3 + x^5 + x^7 + x^9 + \dots$$



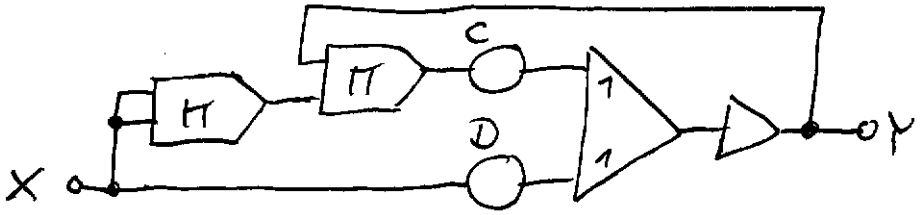
$$Y = \frac{x^3}{1-x} = x^3 + x^4 + x^5 + x^6 + \dots$$



$$Y = \frac{D x^2}{1 - C x^2} = D x^2 + DC x^4 + DC^2 x^6 + DC^3 x^8 + \dots$$

$C = D = 0{,}5$

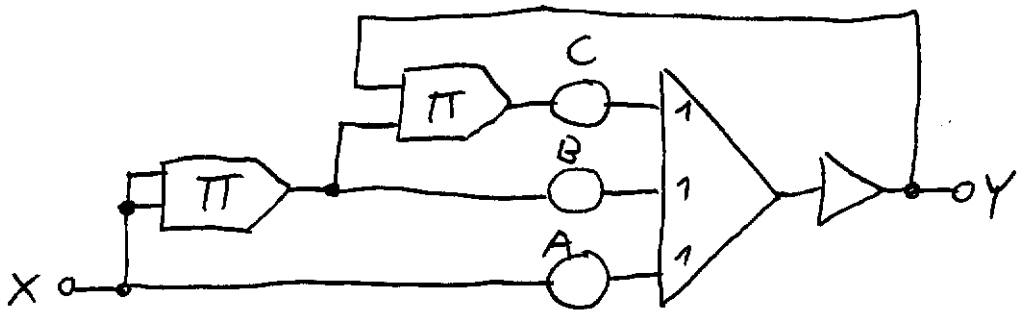
X	Y
-1	1
-0,5	0,1428
0	0
0,5	0,1428
1	1



$$Y = \frac{Dx}{1 - Cx^2} = Dx + DCx^3 + DC^2 x^5 + DC^3 x^7 + \dots$$

$C = D = 1$

X	Y
-0,618	-1
-0,5	-0,666
-0,25	-0,266
0	0
0,25	0,266
0,5	0,666
0,618	1



$$Y = \frac{AX + BX^2}{1 - CX^2} = AX + BX^2 + ACX^3 + BCX^4 + AC^2X^5 + BC^2X^6 + AC^3X^7 + BC^3X^8 + \ldots$$

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + x^4 - \ldots$$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \ldots$$

$$\frac{1}{1+x^2} = 1 - x^2 + x^4 - x^6 + x^8 - \ldots$$

$$\frac{1}{1-x^2} = 1 + x^2 + x^4 + x^6 + x^8 + \ldots$$

$$\frac{1}{1+cx} = 1 - cx + c^2x^2 - c^3x^3 + c^4x^4 - \ldots$$

$$\frac{1}{1-cx} = 1 + cx + c^2x^2 + c^3x^3 + c^4x^4 + \ldots$$

$$\frac{1}{1+cx^2} = 1 - cx^2 + c^2x^4 - c^3x^6 + c^4x^8 - \ldots$$

$$\frac{1}{1-cx^2} = 1 + cx^2 + c^2x^4 + c^3x^6 + c^4x^8 + \ldots$$

$$e^x = 1 + x + x^2/2 + x^3/6 + x^4/24 + x^5/120 + x^6/720 + \ldots$$

4 Hybrid Analog / Digital Computing

Another analog/digital hybrid project: https://www.mdpi.com/1424-8220/23/7/3599

Stochastic DEQs (differential equations) and hybrid computing: https://analogparadigm.com/downloads/alpaca_50.pdf
Source code for Arduino: https://github.com/anabrid/THAThc

4.1 USB Port Specification

A USB-powered device should have at its 5V input a decoupling capacitor between 1 µF and 10 µF.

Source: https://www.beyondlogic.org/usbnutshell/usb2.shtml

I didn't follow this rule in each of my circuits. It should be no problem if the device is already connected to a USB power supply before the supply is powered on.

Don't connect devices with a too large input capacitor to a computer.

4.2 Pinout of THATs Hybrid Port

Wiki: https://the-analog-thing.org/wiki/Hybrid_Computer

Schematic: https://github.com/anabrid/the-analog-thing/blob/main/ANATHING_OSH/ANATHING-BASE%20-%20Schematic5.pdf

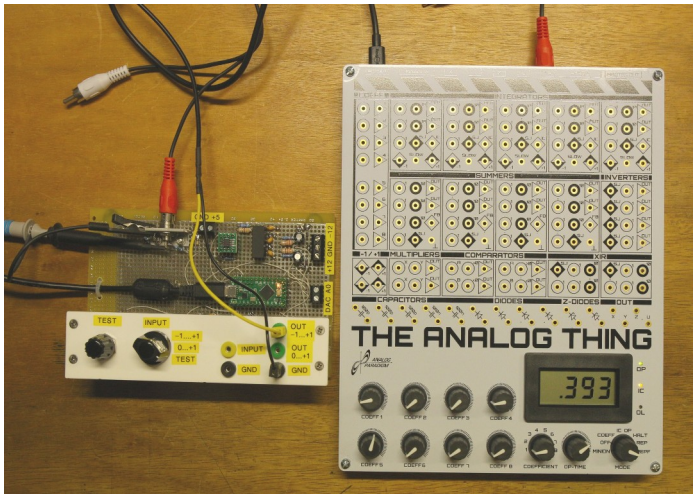
Hybrid port description: https://the-analog-thing.org/w/images/7/78/THAT_hybrid_port_desc.pdf

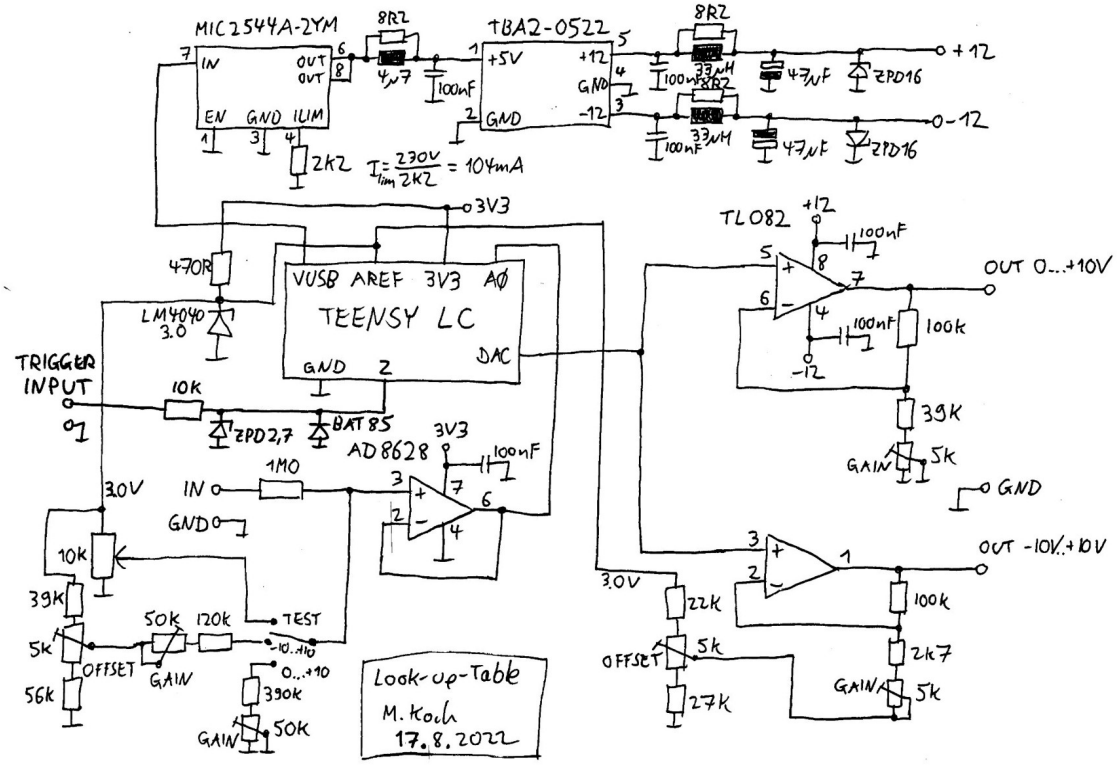
Pin	Name	Function
1 3 5 7	IN-X IN-Y IN-Z IN-U	Same signals as cinch connector X, voltage is 0.1 · OUT-X (OUT-Y, OUT-Z, OUT-U), range -1V to +1V, output impedance is 423 Ohm Might also be used as input, but sinks 21.1 mA at 1 MU, input impedance is 470 Ohm
2 4 6 8	HYB-X HYB-Y HYB-Z HYB-U	Low impedance analog output signals, voltage is 0.1 · OUT-X + 1.6393V, the range is from 0.6393V to 2.6393V for +-1 MU
9, 10	GND	Ground
11, 12	VUSB	Is connected to +5V USB supply voltage via 750mA PTC fuse
13	DIR	This is an input with 3.3V level, low level enables THATs hybrid mode. The input has an internal pullup resistor to 3.3V and it not 5V tolerant.
14	-Mode_OP	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = operate In hybrid mode: 5V tolerant 3.3V input, sets THATs operation mode, the input has no internal pullup resistor
15	Voffset	= 10V / 610 · 100 = 1.6393V
16	-Mode_IC	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = initial condition In hybrid mode: 5V tolerant 3.3V input, sets THATs initial condition mode, the input has no internal pullup resistor

Note: Looking from the rear side into the hybrid port: Pin 1 is top right, Pin 2 is bottom right.

4.3 Teensy LC

This is the schematic diagram of my "Teensy LC" board for hybrid analog / digital computing.





4.3.1 Function Generator with Look-Up-Table

This can be used for any function y=f(x) that can be expressed in C code. http://www.astro-electronic.de/that-look-up-table2.ino

```
// Look-up-table for THAT, realized with Teensy LC
// Michael Koch 2022

boolean in_range = 0;     // 0 means the input range is [0 ... +1] machine units
                          // 1 means the input range is [-1 ... +1] machine units
boolean out_range = 0;    // 0 means the output range is [0 ... +1] machine units
                          // 1 means the output range is [-1 ... +1] machine units
int led = 13;
byte lowbyte[4096];
byte highbyte[2048];

// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT);      // initialize the digital pin as an output

  for(int x=0; x<4096; x+=2)  // fill the look-up-table, two 12-bit entries are compressed into 3 bytes
  {
    int y0 = mu2int(function(int2mu(x)));
    int y1 = mu2int(function(int2mu(x+1)));
    lowbyte[x] = y0 & 0x00ff;
    lowbyte[x+1] = y1 & 0x00ff;
    highbyte[x>>1] = ((y0 & 0x0f00) >> 8) | ((y1 & 0x0f00) >> 4);
  }

  analogReference(EXTERNAL);   // 3.0V reference
  analogReadResolution(12);    // set to 12 bits
  analogWriteResolution(12);   // set to 12 bits

  //  Serial.begin(38400);
}

float function(float x)   // This is the user-defined function with input and output in machine units
{
```

```
  float y = x;
  if ((x>=0.35) && (x<0.45)) y = 0.0;
  if ((x>=0.45) && (x<0.55)) y = 0.5;
  if ((x>=0.55) && (x<0.65)) y = 1.0;
  return(y);
}

float int2mu(int i)       // convert from 12-bit integer to machine units
{
  if (in_range == 0)          // range is [0 ... +1] machine units
    return((float)i / 4096);
  else                        // range is [-1 ... +1] machine units
    return((float)(i - 2048) / 2048);
}

int mu2int(float f)       // convert from machine units to 12-bit integer, and clip to the allowed range
{
  if (out_range == 0)         // range is [0 ... +1] machine units
  {
    if (f < 0) f = 0;
    if (f > 0.9999) f = 0.9999;
    return((int)(f * 4096));
  }
  else                        // range is [-1 ... +1] machine units
  {
    if (f < -1) f = -1;
    if (f > 0.9999) f = 0.9999;
    return(2048 + (int)(f * 2048));
  }
}

void loop() {
  digitalWrite(led, HIGH);   // switch on the LED (for checking the sample rate, about 57kHz)
  int x = analogRead(0);
  digitalWrite(led, LOW);    // switch off the LED
  if(x & 0x0001)
    analogWrite(A12, lowbyte[x] + ((highbyte[x>>1] & 0xf0) << 4));
  else
    analogWrite(A12, lowbyte[x] + ((highbyte[x>>1] & 0x0f) << 8));
```

```
}
```

A problem appears when you try to use the exponential function exp() in the above example. Obviously it needs too much RAM. You can use the following quick-and-dirty power series approximation instead. This is not the most efficient way to approximate the exponential function.

```
float expo(float x)   // This is a power series approximation for the exponential function,
                      // because the built-in exponential function needs too much RAM
{
  float y = 1;
  float z = 1;
  float n = 1;
  for (int i = 1; i < 20; i++)
  {
    z = z * x;
    n = n * i;
    y = y + z / n;
  }
  return(y);
}
```

4.3.2 Look-Up-Table with Derivative

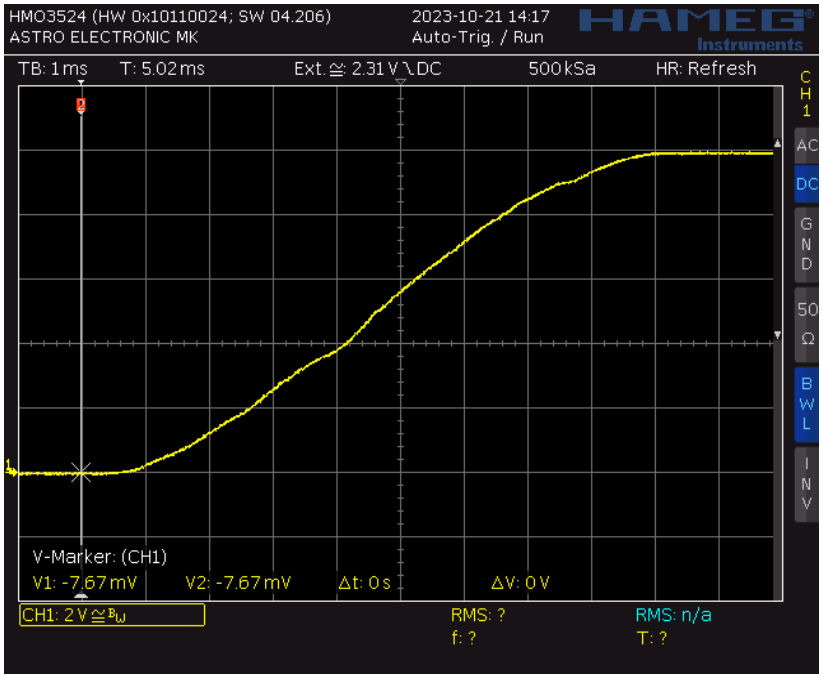
In this software a user-defined table is used, which has less entries (351) than the final look-up-table (4096).

In this example the user-defined table contains the height profile of the Eichelnkopf mountain with 351 points, which are separated by 1 m horizontal distance each. The height difference from start to end is 53.2 m. The maximum slope is about 34-35%.

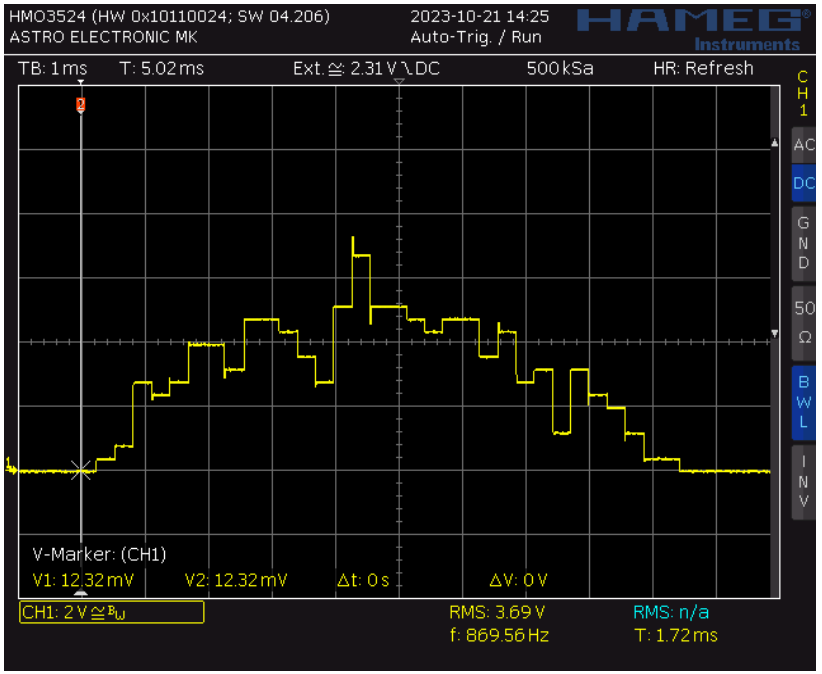
The output of the look-up-table can be either the height profile of its derivative (which is the slope).

http://www.astro-electronic.de/that_look_up_table_derivative.ino

This is the height profile of the Eichelnkopf mountain, 53.2m elevation is equivalent to 1 machine unit (10V):



This is the slope of the Eichelnkopf mountain, 50% slope is equivalant to 1 machine unit (10V):



```
// Look-up-table with derivative for THAT, realized with Teensy LC
// Michael Koch 2023

boolean in_range = 0;    // 0 means the input range is [0 ... +1] machine units
                         // 1 means the input range is [-1 ... +1] machine units
boolean out_range = 0;   // 0 means the output range is [0 ... +1] machine units
                         // 1 means the output range is [-1 ... +1] machine units
int led = 13;
byte lowbyte[4096];
byte highbyte[2048];


// This is the height profile of the Eichelnkopf mountain.
// It consists of 351 points which are separated by 1 m horizontal distance each.
// The height difference from start to end is 53.2 m.
// The maximum slope is about 34-35%.
//
// Input of look-up-table: 350m horizontal distance is equivalent to 1 machine unit.
// Output of look-up-table, in elevation mode: 53.2m or 100m elevation is equivalent to 1 machine unit.
// Output of look-up-table, in slope mode: 35% or 50% slope is equivalent to 1 machine unit.

const int number_of_points = 351;
const float curve[number_of_points] =
{
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.02, 0.04, 0.06, 0.08, 0.1, 0.12, 0.14, 0.16, 0.18, 0.2, 0.24, 0.28, 0.32, 0.36,
  0.4, 0.44, 0.48, 0.52, 0.56, 0.6, 0.74, 0.88, 1.02, 1.16, 1.3, 1.44, 1.58, 1.72, 1.86, 2, 2.12, 2.24, 2.36, 2.48,
  2.6, 2.72, 2.84, 2.96, 3.08, 3.2, 3.34, 3.48, 3.62, 3.76, 3.9, 4.04, 4.18, 4.32, 4.46, 4.6, 4.8, 5, 5.2, 5.4, 5.6,
  5.8, 6, 6.2, 6.4, 6.6, 6.8, 7, 7.2, 7.4, 7.6, 7.8, 8, 8.2, 8.4, 8.6, 8.76, 8.92, 9.08, 9.24, 9.4, 9.56, 9.72, 9.88,
  10.04, 10.2, 10.44, 10.68, 10.92, 11.16, 11.4, 11.64, 11.88, 12.12, 12.36, 12.6, 12.84, 13.08, 13.32, 13.56, 13.8,
  14.04, 14.28, 14.52, 14.76, 15, 15.22, 15.44, 15.66, 15.88, 16.1, 16.32, 16.54, 16.76, 16.98, 17.2, 17.38, 17.56,
  17.74, 17.92, 18.1, 18.28, 18.46, 18.64, 18.82, 19, 19.14, 19.28, 19.42, 19.56, 19.7, 19.84, 19.98, 20.12, 20.26,
  20.4, 20.66, 20.92, 21.18, 21.44, 21.7, 21.96, 22.22, 22.48, 22.74, 23, 23.34, 23.68, 24.02, 24.36, 24.7, 25.04,
  25.38, 25.72, 26.06, 26.4, 26.66, 26.92, 27.18, 27.44, 27.7, 27.96, 28.22, 28.48, 28.74, 29, 29.26, 29.52, 29.78,
  30.04, 30.3, 30.56, 30.82, 31.08, 31.34, 31.6, 31.84, 32.08, 32.32, 32.56, 32.8, 33.04, 33.28, 33.52, 33.76, 34,
  34.22, 34.44, 34.66, 34.88, 35.1, 35.32, 35.54, 35.76, 35.98, 36.2, 36.44, 36.68, 36.92, 37.16, 37.4, 37.64, 37.88,
  38.12, 38.36, 38.6, 38.84, 39.08, 39.32, 39.56, 39.8, 40.04, 40.28, 40.52, 40.76, 41, 41.18, 41.36, 41.54, 41.72,
  41.9, 42.08, 42.26, 42.44, 42.62, 42.8, 43.02, 43.24, 43.46, 43.68, 43.9, 44.12, 44.34, 44.56, 44.78, 45, 45.14,
  45.28, 45.42, 45.56, 45.7, 45.84, 45.98, 46.12, 46.26, 46.4, 46.56, 46.72, 46.88, 47.04, 47.2, 47.36, 47.52, 47.68,
  47.84, 48, 48.06, 48.12, 48.18, 48.24, 48.3, 48.36, 48.42, 48.48, 48.54, 48.6, 48.76, 48.92, 49.08, 49.24, 49.4,
```

```
  49.56, 49.72, 49.88, 50.04, 50.2, 50.32, 50.44, 50.56, 50.68, 50.8, 50.92, 51.04, 51.16, 51.28, 51.4, 51.5, 51.6,
  51.7, 51.8, 51.9, 52, 52.1, 52.2, 52.3, 52.4, 52.46, 52.52, 52.58, 52.64, 52.7, 52.76, 52.82, 52.88, 52.94, 53,
  53.02, 53.04, 53.06, 53.08, 53.1, 53.12, 53.14, 53.16, 53.18, 53.2, 53.22, 53.24, 53.26, 53.28, 53.3, 53.32, 53.34,
  53.36, 53.38, 53.4, 53.38, 53.36, 53.34, 53.32, 53.3, 53.28, 53.26, 53.24, 53.22, 53.2, 53.2, 53.2, 53.2, 53.2,
  53.2, 53.2, 53.2, 53.2, 53.2, 53.2
};

// the setup routine runs once when you press reset:
void setup()
{
  pinMode(led, OUTPUT);        // initialize the digital pin as an output
  analogReference(EXTERNAL);   // 3.0V reference
  analogReadResolution(12);    // set to 12 bits
  analogWriteResolution(12);   // set to 12 bits

  for(int x=0; x<4096; x+=2)   // fill the look-up-table, two 12-bit entries are compressed into 3 bytes
  {
    int y0 = mu2int(function(int2mu(x)));
    int y1 = mu2int(function(int2mu(x+1)));
    lowbyte[x] = y0 & 0x00ff;
    lowbyte[x+1] = y1 & 0x00ff;
    highbyte[x>>1] = ((y0 & 0x0f00) >> 8) | ((y1 & 0x0f00) >> 4);
  }
}

float function(float x)   // This is the user-defined function with input and output in machine units
{
  int index = int(x * number_of_points);
  int index2 = index+1;
  if (index2 == number_of_points) index2--;
  //float y = curve[index] / 53.2;                    // use this line for elevation as output, 53.2m = 1 machine unit
  //float y = curve[index] / 100;                     // use this line for elevation as output, 100m = 1 machine unit
  //float y = (index2] - curve[index]) / 0.35;        // use this line for slope as output, 35% slope = 1 machine unit
  float y = (curve[index2] - curve[index]) / 0.5;   // use this line for slope as output, 50% slope = 1 machine unit
  return(y);
}

float int2mu(int i)       // convert from 12-bit integer to machine units
{
```

```
  if (in_range == 0)          // range is [0 ... +1] machine units
    return((float)i / 4096);
  else                        // range is [-1 ... +1] machine units
    return((float)(i - 2048) / 2048);
}

int mu2int(float f)       // convert from machine units to 12-bit integer, and clip to the allowed range
{
  if (out_range == 0)         // range is [0 ... +1] machine units
  {
    if (f < 0) f = 0;
    if (f > 0.9999) f = 0.9999;
    return((int)(f * 4096));
  }
  else                        // range is [-1 ... +1] machine units
  {
    if (f < -1) f = -1;
    if (f > 0.9999) f = 0.9999;
    return(2048 + (int)(f * 2048));
  }
}

void loop()
{
  digitalWrite(led, HIGH);   // switch on the LED (for checking the sample rate, about 57kHz)
  int x = analogRead(0);
  digitalWrite(led, LOW);    // switch off the LED
  if(x & 0x0001)
    analogWrite(A12, lowbyte[x] + ((highbyte[x>>1] & 0xf0) << 4));
  else
    analogWrite(A12, lowbyte[x] + ((highbyte[x>>1] & 0x0f) << 8));
}
```

4.3.3 Delay Line

http://www.astro-electronic.de/that-delay-line.ino

```
// Delay line for THAT, realized with Teensy LC
// Michael Koch 2022

int led = 13;
short buffer[3072];
int buffer_length;
int pointer = 0;
IntervalTimer myTimer;

// the setup routine runs once when you press reset:
void setup()
{
  pinMode(led, OUTPUT);         // initialize the digital pin as an output
  analogReference(EXTERNAL);    // 3.0V reference
  analogReadResolution(12);     // set ADC to 12 bits
  analogWriteResolution(12);    // set DAC to 12 bits

  // delay_time = buffer_length * sample_interval
  buffer_length = 1000;                  // set the length of the buffer (must be <= 3072)
  myTimer.begin(timer_interrupt, 1000);  // set the sample interval in microseconds (must be >= 20)
  myTimer.priority(0);                   // high priority for timer interrupt
}

void timer_interrupt()
{
  digitalWrite(led, HIGH);              // switch on the LED (for checking the sample rate)
  analogWrite(A12, buffer[pointer]);    // write to DAC
  buffer[pointer] = analogRead(A0);     // read from ADC
  if (++pointer == buffer_length)       // increment pointer
    pointer = 0;
  digitalWrite(led, LOW);               // switch off the LED
}
```

```
void loop()   // nothing to do in the main loop
{
}
```

In this paper from 1991 Hans Moravec describes what could be done with hypothetical negative delay time elements:
http://reprints.gravitywaves.com/TimeTravel/Moorevec-1991_TimeTravelAndComputing.pdf

4.3.4 Direct Digital Synthesis

Create any periodic waveform with any floating-point frequency: http://www.astro-electronic.de/that-direct-digital-synthesis.ino

```
// Direct digital synthesis for THAT, realized with Teensy LC
// Michael Koch 2022

boolean out_range = 1;   // 0 means the output range is [0 ... +1] machine units
                         // 1 means the output range is [-1 ... +1] machine units
float freq = 440;        // Output frequency in Hz

int led = 13;
short waveform[2048];
unsigned int ph;
unsigned int dp;
IntervalTimer myTimer;

void setup()    // the setup routine runs once when you press reset
{
  pinMode(led, OUTPUT);        // initialize the digital pin as an output
  analogReference(EXTERNAL);   // 3.0V reference
  analogWriteResolution(12);   // set DAC to 12 bits

  float pi = 4 * atan(1.0);    // calculate pi
  for(int x=0; x<2048; x++)    // fill the waveform array
    waveform[x] = mu2int(function(x / 2048.0 * 2 * pi));

  dp = (unsigned int)(freq * 65536.0 * 65536.0 / 125000.0);  // calculate the value that is added to the 32-bit phase
accumulator

  myTimer.begin(timer_interrupt, 8);  // set the sample interval to 8 microseconds (125kHz)
  myTimer.priority(0);                // high priority for timer interrupt
}

float function(float x)   // this function contains the user-defined waveform
{                         // x is in the [0..2*pi] range and the output is in machine units.
  float y = 0.7 * sin(x) + 0.2 * sin(3*x) + 0.1 * sin(5*x);   // sine with 3rd and 5th harmonic
```

```
  if (x < 0.05) y += 0.5;                                    // add a short impulse
  return(y);
}

int mu2int(float f)       // convert from machine units to 12-bit integer, and clip to the allowed range
{
  if (out_range == 0)         // range is [0 ... +1] machine units
  {
    if (f < 0) f = 0;
    if (f > 0.9999) f = 0.9999;
    return((int)(f * 4096));
  }
  else                        // range is [-1 ... +1] machine units
  {
    if (f < -1) f = -1;
    if (f > 0.9999) f = 0.9999;
    return(2048 + (int)(f * 2048));
  }
}

void timer_interrupt()
{
  digitalWrite(led, HIGH);                // switch on the LED (for checking the sample rate)
  ph += dp;                               // update the phase accumulator
  analogWrite(A12, waveform[ph >> 21]);   // use the most significant 11 bits as index for waveform array, and write to
DAC
  digitalWrite(led, LOW);                 // switch off the LED
}

void loop()   // nothing to do in the main loop
{
}
```

4.3.5 Voltage Controlled Direct Digital Synthesis

Same as before, but the frequency can be adjusted with the analog input voltage.

http://www.astro-electronic.de/that-vco-direct-digital-synthesis.ino

```
// VCO with direct digital synthesis for THAT, realized with Teensy LC
// Michael Koch 2022

boolean out_range = 1;   // 0 means the output range is [0 ... +1] machine units
                         // 1 means the output range is [-1 ... +1] machine units
boolean vco_enable = 1;  // 0 means the output frequency is constant
                         // 1 means the output frequency is proportional to the input voltage, from 0 to freq
float freq = 880.0;      // Output frequency in Hz

int led = 13;
short waveform[2048];
unsigned int ph = 0;
unsigned int dp = 0;
IntervalTimer myTimer1;
IntervalTimer myTimer2;

void setup()    // the setup routine runs once when you press reset
{
  pinMode(led, OUTPUT);         // initialize the digital pin as an output
  analogReference(EXTERNAL);    // 3.0V reference
  analogReadResolution(12);     // set ADC to 12 bits
  analogWriteResolution(12);    // set DAC to 12 bits

  float pi = 4 * atan(1.0);     // calculate pi
  for(int x=0; x<2048; x++)     // fill the waveform array
    waveform[x] = mu2int(function(x / 2048.0 * 2 * pi));

  if (vco_enable == 1)
    myTimer1.begin(update_frequency, 1000);  // set the frequency update interval to 1000 microseconds (1kHz)
  else
    dp = 34359.738 * freq;    // calculate the value that is added to the 32-bit phase accumulator
```

```
                              //  2^32 / 125000 = 34359.738

  myTimer2.begin(timer_interrupt, 8);  // set the sample interval to 8 microseconds (125kHz)
  myTimer2.priority(0);                // high priority for timer interrupt
}

float function(float x)   // this function contains the user-defined waveform
{                         // x is in the [0..2*pi] range and the output is in machine units.
  float y = 0.7 * sin(x) + 0.2 * sin(3*x) + 0.1 * sin(5*x);   // sine with 3rd and 5th harmonic
  return(y);
}

int mu2int(float f)       // convert from machine units to 12-bit integer, and clip to the allowed range
{
  if (out_range == 0)         // range is [0 ... +1] machine units
  {
    if (f < 0) f = 0;
    if (f > 0.9999) f = 0.9999;
    return((int)(f * 4096));
  }
  else                        // range is [-1 ... +1] machine units
  {
    if (f < -1) f = -1;
    if (f > 0.9999) f = 0.9999;
    return(2048 + (int)(f * 2048));
  }
}

void update_frequency()
{
  digitalWrite(led, HIGH);                 // switch on the LED
  dp = analogRead(A0) * 8.3906565 * freq;  // calculate the value that is added to the 32-bit phase accumulator
                                           //  2^32 / 125000 / 4095 = 8.3906565
  digitalWrite(led, LOW);                  // switch off the LED
}

void timer_interrupt()
{
  ph += dp;                               // update the phase accumulator
```

```
  analogWrite(A12, waveform[ph >> 21]);   // use the most significant 11 bits as index for array, and write to DAC
}

void loop()   // nothing to do in the main loop
{
}
```

4.3.6 Triggered Signal Generators

Synchronized with THAT's trigger signal

http://www.astro-electronic.de/that-triggered-signal-generator.ino

```
// Triggered signal generator for THAT, realized with Teensy LC
// Michael Koch 2022

boolean out_range = 1;        // 0 means the output range is [0...+1] machine units
                              // 1 means the output range is [-1...+1] machine units

boolean mode = 0;             // 0 means one-shot operation
                              // 1 means loop operation

int sample_interval = 10;     // set the sample interval in microseconds, must be >= 10
                              // the total duration of the waveform is 3600 * sample_interval
                              // Examples:
                              // sample_interval = 10      duration = 36ms
                              // sample_interval = 100     duration = 360ms
                              // sample_interval = 1000    duration = 3.6s
                              // sample_interval = 10000   duration = 36s
int led = 13;
int trig = 2;
byte lowbyte[3600];
byte highbyte[1800];
IntervalTimer myTimer;
int x;

void setup()        // the setup routine runs once when you press reset
{
  pinMode(led, OUTPUT);         // initialize the digital pin as an output
  pinMode(trig, INPUT);         // initialize the digital pin as an input
  analogReference(EXTERNAL);    // 3.0V reference
  analogWriteResolution(12);    // set to 12 bits

  for(x=0; x<3600; x+=2)        // fill the array with the waveform, two entries are compressed into 3 bytes
```

```
  {
    float t0 = x * (float)sample_interval * 1e-6;       // time in seconds
    float t1 = (x+1) * (float)sample_interval * 1e-6;
    int y0 = mu2int(function(t0));
    int y1 = mu2int(function(t1));
    lowbyte[x] = y0 & 0x00ff;
    lowbyte[x+1] = y1 & 0x00ff;
    highbyte[x>>1] = ((y0 & 0x0f00) >> 8) | ((y1 & 0x0f00) >> 4);
  }

  myTimer.begin(timer_interrupt, sample_interval);  // start the timer interrupt
  myTimer.priority(0);                              // high priority for timer interrupt
}

float function(float t)   // This is the user-defined waveform with input t in seconds and output in machine units
{
  float y = exp(-50 * t) * sin(2 * 3.1415 * 440 * t);   // 440Hz sine with exponential damping
  return(y);
}

int mu2int(float f)       // convert from machine units to 12-bit integer, and clip to the allowed range
{
  if (out_range == 0)         // range is [0 ... +1] machine units
  {
    if (f < 0) f = 0;
    if (f > 0.9999) f = 0.9999;
    return((int)(f * 4096));
  }
  else                        // range is [-1 ... +1] machine units
  {
    if (f < -1) f = -1;
    if (f > 0.9999) f = 0.9999;
    return(2048 + (int)(f * 2048));
  }
}

void timer_interrupt()
{
  if (digitalRead(trig) == HIGH)    // THAT is in initial condition
```

```
  {
    digitalWrite(led, HIGH);        // switch on the LED
    x = 0;                          // in initial condition the first element of the waveform array is written to DAC
  }
  else                              // THAT is in operate mode
  {
    digitalWrite(led, LOW);         // switch off the LED
    x++;                            // increase the pointer
    if (x >= 3600)                  // is the end of the waveform array reached?
    {
      if (mode == 0)
        x = 3599;                   // in one-shot mode, the pointer stays at the last element
      else
        x = 0;                      // in loop mode, the pointer jumps to the first element
    }
  }
  if(x & 0x0001)                    // write value to DAC
    analogWrite(A12, lowbyte[x] + ((highbyte[x>>1] & 0xf0) << 4));
  else
    analogWrite(A12, lowbyte[x] + ((highbyte[x>>1] & 0x0f) << 8));
}

void loop()   // nothing to do in the main loop
{
}
```

4.3.7 Band-Limited Noise Generator

http://www.astro-electronic.de/that-band_limited_noise.ino

```
// Band limited noise generator for THAT, realized with Teensy LC
// Michael Koch 2023

float samplerate = 10000;  // Sample rate in Hz, 1000000/samplerate should be an integer,
                           //  do not set higher than about 125000 Hz
float freq1 = 400;         // Lower band limit in Hz
float freq2 = 600;         // Upper band limit in Hz

const int length = 3000;          // length of waveform table,
                                  //  do not set higher than about 3000
float df = samplerate / length;   // distance of lines in spectrum in Hz
short waveform[length];           // waveform table
float pi = 4 * atan(1.0);         // calculate pi
int i = 0;                        // index counter
int led = 13;                     // LED output
IntervalTimer myTimer;

void setup()    // the setup routine runs once when you press reset
{
  pinMode(led, OUTPUT);        // initialize the digital pin as an output
  analogReference(EXTERNAL);   // 3.0V reference
  analogWriteResolution(12);   // set DAC to 12 bits

  for(int x = 0; x < length; x++)    // fill the waveform array
    waveform[x] = 2048 + (int)(2048 * function(x));

  myTimer.begin(timer_interrupt, (int)(1000000 / samplerate));  // set the interrupt interval
  myTimer.priority(0);                  // high priority for timer interrupt
}

float function(int x)   // calculate band-limited sum of sine waves
```

```
{
  float y = 0;
  float ph;
  randomSeed(13);       // use always the same seed for reproducible random
  for(float f = freq1; f <= freq2; f += df)
  {
    ph = 0.001 * (float)random(6283);                   // random phase [0...2*pi]
    y += sin(ph + (float)x * 2 * pi * f / samplerate);  // sum of many sine waves
  }
  y = y * 3.0 * df / (freq2 - freq1);                   // normalize to avoid clipping
  return(y);
}

void timer_interrupt()
{
  digitalWrite(led, HIGH);          // switch on the LED (only for checking the sample rate)
  if (++i >= length) i = 0;         // increment index counter
  analogWrite(A12, waveform[i]);    // read from the waveform array and write to DAC
  digitalWrite(led, LOW);           // switch off the LED
}

void loop()   // nothing to do in the main loop
{
}
```

4.3.8 Band-Limited Noise Generator with a Control-Voltage Input

Control voltage from synthesizer (1V / octave, Arturia MatrixBrute):

http://www.astro-electronic.de/that-noise-with-cv.ino

```
// Noise generator with control voltage input, realized with Teensy LC
// Michael Koch 2023

float samplerate = 10000;          // Sample rate in Hz (for 1kHz)
const int length = 2500;           // length of waveform table,
                                   //  do not set higher than about 2800
float df = samplerate / length;    // distance of lines in spectrum in Hz
float freq1 = 900;          // Lower band limit in Hz, center frequency should be 1kHz
                            //  freq1 should be a multiple of df
float freq2 = 1110;         // Upper band limit in Hz, center frequency should be 1kHz


short waveform[length];            // waveform table
float pi = 4 * atan(1.0);          // calculate pi
int i = 0;                         // index counter
int led = 13;                      // LED output
IntervalTimer myTimer1, myTimer2;

void setup()    // the setup routine runs once when you press reset
{
  pinMode(led, OUTPUT);         // initialize the digital pin as an output
  analogReference(EXTERNAL);    // 3.0V reference
  analogWriteResolution(12);    // set DAC to 12 bits
  analogReadResolution(12);     // set ADC to 12 bits

  for(int x = 0; x < length; x++)    // fill the waveform array
  {
    waveform[x] = 2048 + (int)(2048 * function(x));
    if (x % 32 < 16)
      digitalWrite(led, LOW);     // switch off the LED as a progress indicator
    else
```

```
      digitalWrite(led, HIGH);    // switch on the LED as a progress indicator
  }

  myTimer1.begin(update_frequency, 10000);  // set the frequency update interval to 10 milliseconds
  myTimer1.priority(20);                    // lower priority for timer interrupt

  myTimer2.begin(timer_interrupt, (int)(1000000 / samplerate));  // set the interrupt interval
  myTimer2.priority(0);                  // high priority for timer interrupt
}

float function(int x)   // calculate band-limited sum of sine waves
{
  float y = 0;
  float ph;
  randomSeed(13);       // use always the same seed for reproducible random
  for(float f = freq1; f <= freq2; f += df)
  {
    ph = 0.001 * (float)random(6283);                    // random phase [0...2*pi]
    y += sin(ph + (float)x * 2 * pi * f / samplerate);   // sum of many sine waves
  }
  y = y * 2.5 * df / (freq2 - freq1);                    // normalize to avoid clipping
  return(y);
}

void update_frequency()
{
  float cv = 10.0 / 4096.0 * analogRead(A0);    // read the control voltage [0V...10V]
  float freq = 440 * pow(2, cv - 4.75);         // frequency = 440Hz * 2^(CV â^’ 4.75V)
                                                //  for Arturia Matrixbrute
  if (freq > 12500) freq = 12500;               // allowed frequency range is 1Hz to 12.5kHz
  if (freq < 1) freq = 1;
  myTimer2.update((int)(100000.0 / freq));
}

void timer_interrupt()
{
  digitalWrite(led, HIGH);          // switch on the LED (only for checking the sample rate)
  if (++i >= length) i = 0;         // increment index counter
  analogWrite(A12, waveform[i]);    // read from the waveform array and write to DAC
```

```
  digitalWrite(led, LOW);           // switch off the LED
}

void loop()   // nothing to do in the main loop
{
}
```

4.3.9 Heart Sound Synthesis with a Control Voltage Input

Let's assume you have an audio waveform that you want to insert into a C source code. That requires to convert the waveform into a comma separated list of hexadecimal numbers. For example I wanted to insert a heart sound wave (with length 2048 samples) into the C source code for a microcontroller.

I found a suitable *.wav file here: https://en.wikipedia.org/wiki/File:Emily's_heartbeat.wav

Step 1:
Use Sonic Visualiser to find the start and end points of a segment, which equals exactly one heart beat: Start = 8.183s, End = 9.128s. Also check the maximum amplitude of the signal within this segment. Theoretically this could also be done in FFmpeg, but something was broken at the moment.
Sonic Visualiser: https://www.sonicvisualiser.org

Step 2:
Use FFmpeg to cut out the segment, change the sample rate and amplify the signal, so that the maximum level is just below clipping level.
FFmpeg: https://ffmpeg.org// FFmpeg binaries for Windows: https://www.gyan.dev/ffmpeg/builds/

This batch file makes the conversion:

```
set "START=8.183"    // Start of segment in seconds
set "STOP=9.128"     // End of segment in seconds
set "VOL=1.56"       // Volume, choose so that signal isn't clipping
set "SAMPLES=2048"   // Number of audio samples

ffmpeg -i "Emily's_heartbeat.wav" -ss %START% -to %STOP% -af aresample='%SAMPLES%/(%STOP%-%START%)',volume=%VOL% -ac 1 -
y out.wav

pause
```

Step 3:
Open the file out.wav in Sonic Visualiser and export as CSV file: File --> Export Audio to Data File

Step 4:
Open the file in Libre Office Calc. Column A contains the sample number (from 1 to 2048) and column B contains the signal values (from -1 to +1).
Use this function in field C1 for converting to 12-bit unsigned integer, written as a hexadecimal number in C syntax:
`=CONCAT("0X",DEC2HEX(INT(2048+B1*2048)))`
Select field C1. There is a small dot in the lower right corner of this field. Draw this dot downwards to copy the content to all 2047 other fields below.
Now use this function in field D1 to concat all numbers together into one field, separated by a comma and a space character:
`=TEXTJOIN(", ",1,C1:C2048)`

Now copy and paste the content of this field to the C source code where you need it.

```
// Heart Sound Synthesis for THAT, realized with Teensy LC
// Michael Koch 2024

float freq = 240.0;         // heart beats per minute

int led = 13;
static const unsigned short waveform[] =
{
 0X7FF, 0X7FE, 0X7FD, 0X7FF, 0X801, 0X801, 0X7FF, 0X7FD, 0X7FA, 0X7F9, 0X7FA, 0X7FC, 0X800, 0X803, 0X804, 0X805,
 0X806, 0X805, 0X803, 0X7FF, 0X7FB, 0X7F8, 0X7F5, 0X7F1, 0X7F0, 0X7EF, 0X7EF, 0X7EE, 0X7EC, 0X7E8, 0X7E6, 0X7E3,
 0X7E2, 0X7E0, 0X7D9, 0X7D3, 0X7D1, 0X7D2, 0X7D4, 0X7D8, 0X7E0, 0X7EB, 0X7FB, 0X80C, 0X81D, 0X828, 0X82F, 0X833,
 0X835, 0X834, 0X831, 0X82C, 0X828, 0X824, 0X822, 0X822, 0X823, 0X825, 0X826, 0X826, 0X825, 0X824, 0X823, 0X823,
 0X820, 0X81C, 0X817, 0X813, 0X813, 0X814, 0X816, 0X817, 0X815, 0X812, 0X80E, 0X80E, 0X812, 0X819, 0X821, 0X827,
 0X82C, 0X82E, 0X82D, 0X828, 0X821, 0X817, 0X807, 0X7F4, 0X7DF, 0X7CB, 0X7B8, 0X7A9, 0X79D, 0X796, 0X793, 0X793,
 0X793, 0X792, 0X791, 0X78F, 0X790, 0X793, 0X799, 0X7A2, 0X7AB, 0X7B4, 0X7BB, 0X7BE, 0X7C2, 0X7C7, 0X7CD, 0X7D2,
 0X7D5, 0X7D6, 0X7D5, 0X7CF, 0X7C6, 0X7BB, 0X7AD, 0X7A0, 0X797, 0X794, 0X797, 0X7A0, 0X7AA, 0X7B1, 0X7B8, 0X7C1,
 0X7CF, 0X7E3, 0X7FE, 0X81D, 0X83E, 0X860, 0X881, 0X8A2, 0X8C7, 0X8EC, 0X90D, 0X928, 0X93F, 0X955, 0X96E, 0X98D,
 0X9A8, 0X9B3, 0X9A7, 0X98E, 0X97B, 0X97A, 0X98B, 0X99F, 0X99D, 0X984, 0X964, 0X94A, 0X928, 0X8D4, 0X821, 0X713,
 0X5E3, 0X4D7, 0X420, 0X3D6, 0X3E4, 0X40A, 0X402, 0X3BA, 0X367, 0X351, 0X39D, 0X442, 0X51A, 0X5F2, 0X6A5, 0X736,
 0X7CA, 0X871, 0X924, 0X9D1, 0XA67, 0XADE, 0XB38, 0XB7D, 0XBB6, 0XBE3, 0XC00, 0XC06, 0XBF3, 0XBC6, 0XB81, 0XB22,
 0XAAA, 0XA1C, 0X985, 0X8F5, 0X87B, 0X826, 0X802, 0X809, 0X82F, 0X868, 0X8A7, 0X8DC, 0X8FD, 0X904, 0X8F6, 0X8D9,
 0X8AF, 0X879, 0X837, 0X7EC, 0X79F, 0X759, 0X726, 0X70C, 0X70A, 0X71F, 0X73D, 0X757, 0X765, 0X765, 0X756, 0X72E,
 0X6EC, 0X697, 0X63E, 0X5F3, 0X5B4, 0X56B, 0X515, 0X4BF, 0X478, 0X451, 0X44A, 0X457, 0X462, 0X453, 0X435, 0X427,
 0X427, 0X414, 0X3F1, 0X3D9, 0X3D2, 0X3F4, 0X448, 0X4B9, 0X557, 0X644, 0X767, 0X897, 0X9C2, 0XADF, 0XBEE, 0XCFF,
 0XE12, 0XF0B, 0XFB5, 0XFEF, 0XFC6, 0XF57, 0XEC1, 0XE1A, 0XD66, 0XCA8, 0XBE2, 0XB19, 0XA5A, 0X9B2, 0X923, 0X8A8,
 0X83A, 0X7D7, 0X77F, 0X736, 0X702, 0X6E3, 0X6D5, 0X6D4, 0X6DB, 0X6EC, 0X702, 0X71B, 0X737, 0X751, 0X765, 0X774,
 0X781, 0X78E, 0X79B, 0X7A6, 0X7B1, 0X7BA, 0X7BE, 0X7BA, 0X7B1, 0X7A6, 0X7A0, 0X79E, 0X79E, 0X796, 0X786, 0X76B,
 0X74F, 0X738, 0X725, 0X716, 0X706, 0X6F1, 0X6DC, 0X6CB, 0X6C0, 0X6BC, 0X6C0, 0X6C7, 0X6D1, 0X6DD, 0X6EB, 0X6FA,
 0X70A, 0X719, 0X726, 0X733, 0X742, 0X751, 0X761, 0X770, 0X77F, 0X78C, 0X798, 0X7A5, 0X7B1, 0X7BE, 0X7CD, 0X7DE,
```

```
0X7F2, 0X808, 0X81F, 0X836, 0X849, 0X857, 0X861, 0X86A, 0X873, 0X87C, 0X886, 0X88D, 0X893, 0X898, 0X89C, 0X8A2,
0X8AA, 0X8B6, 0X8C4, 0X8D2, 0X8E0, 0X8EB, 0X8F3, 0X8F7, 0X8F6, 0X8F0, 0X8E7, 0X8DE, 0X8D8, 0X8D4, 0X8CE, 0X8C6,
0X8B8, 0X8A3, 0X888, 0X86B, 0X84F, 0X839, 0X82A, 0X81D, 0X810, 0X802, 0X7F1, 0X7E2, 0X7D6, 0X7CE, 0X7C8, 0X7C3,
0X7BE, 0X7BA, 0X7B9, 0X7BA, 0X7BE, 0X7C2, 0X7C4, 0X7C5, 0X7C5, 0X7C5, 0X7C4, 0X7C1, 0X7BA, 0X7B2, 0X7A9, 0X7A2,
0X79F, 0X7A2, 0X7A9, 0X7B3, 0X7BF, 0X7CC, 0X7D8, 0X7E3, 0X7ED, 0X7F4, 0X7FA, 0X7FE, 0X801, 0X802, 0X802, 0X7FC,
0X7F5, 0X7EC, 0X7E2, 0X7D9, 0X7D2, 0X7CD, 0X7C9, 0X7C5, 0X7C2, 0X7BF, 0X7BD, 0X7BB, 0X7BB, 0X7BE, 0X7C5, 0X7CC,
0X7D1, 0X7D3, 0X7D6, 0X7D9, 0X7E0, 0X7E4, 0X7E6, 0X7E4, 0X7E3, 0X7E8, 0X7F5, 0X805, 0X80F, 0X810, 0X80A, 0X802,
0X7FC, 0X7FB, 0X7FB, 0X7FB, 0X7FB, 0X7FB, 0X800, 0X80A, 0X817, 0X822, 0X826, 0X823, 0X81D, 0X81A, 0X820, 0X82E,
0X83D, 0X846, 0X845, 0X83E, 0X833, 0X82C, 0X82A, 0X82D, 0X830, 0X831, 0X82C, 0X824, 0X81B, 0X812, 0X80B, 0X807,
0X802, 0X7FD, 0X7F8, 0X7F4, 0X7F0, 0X7ED, 0X7EB, 0X7EA, 0X7EA, 0X7E9, 0X7E9, 0X7E8, 0X7E7, 0X7E7, 0X7EA, 0X7F0,
0X7F4, 0X7F6, 0X7F7, 0X7F6, 0X7F6, 0X7FA, 0X7FD, 0X800, 0X7FF, 0X7FB, 0X7F3, 0X7ED, 0X7EA, 0X7EC, 0X7EE, 0X7EF,
0X7EE, 0X7ED, 0X7EF, 0X7F3, 0X7FA, 0X801, 0X806, 0X80A, 0X80D, 0X80E, 0X810, 0X814, 0X817, 0X819, 0X819, 0X818,
0X817, 0X817, 0X817, 0X816, 0X814, 0X813, 0X811, 0X811, 0X813, 0X813, 0X814, 0X814, 0X815, 0X815, 0X815, 0X815,
0X813, 0X80F, 0X809, 0X803, 0X7FD, 0X7F8, 0X7F5, 0X7F4, 0X7F3, 0X7F3, 0X7F6, 0X7F9, 0X7FF, 0X806, 0X80D, 0X814,
0X819, 0X81C, 0X81E, 0X81E, 0X81D, 0X81B, 0X818, 0X813, 0X80E, 0X807, 0X801, 0X7FB, 0X7F5, 0X7F1, 0X7EE, 0X7EF,
0X7F0, 0X7F1, 0X7F0, 0X7ED, 0X7EB, 0X7E9, 0X7E8, 0X7E8, 0X7E6, 0X7E2, 0X7DE, 0X7DC, 0X7DC, 0X7DF, 0X7E3, 0X7E9,
0X7ED, 0X7EF, 0X7F1, 0X7F2, 0X7F5, 0X7F8, 0X7FB, 0X7FD, 0X7FC, 0X7FA, 0X7F8, 0X7F5, 0X7F3, 0X7F0, 0X7ED, 0X7EB,
0X7EA, 0X7EA, 0X7EC, 0X7EE, 0X7EF, 0X7F0, 0X7F1, 0X7F1, 0X7F2, 0X7F4, 0X7F5, 0X7F7, 0X7F9, 0X7FB, 0X7FF, 0X802,
0X803, 0X805, 0X805, 0X805, 0X805, 0X803, 0X802, 0X800, 0X7FD, 0X7FD, 0X7FE, 0X7FF, 0X801, 0X802, 0X802, 0X801,
0X802, 0X804, 0X808, 0X80D, 0X810, 0X811, 0X812, 0X813, 0X815, 0X817, 0X818, 0X819, 0X818, 0X817, 0X817, 0X816,
0X812, 0X80E, 0X80C, 0X80D, 0X80F, 0X812, 0X813, 0X814, 0X814, 0X814, 0X812, 0X80F, 0X80B, 0X805, 0X802, 0X7FF,
0X7FC, 0X7FC, 0X7FC, 0X7FC, 0X7FC, 0X7FC, 0X7FC, 0X7FB, 0X7FB, 0X7FB, 0X7FC, 0X7FF, 0X7FF, 0X7FF, 0X800, 0X800,
0X800, 0X800, 0X801, 0X802, 0X802, 0X800, 0X7FE, 0X7FD, 0X7FD, 0X800, 0X803, 0X805, 0X805, 0X803, 0X800, 0X7FD,
0X7FB, 0X7FB, 0X7F9, 0X7F7, 0X7F3, 0X7F1, 0X7EF, 0X7ED, 0X7EE, 0X7EF, 0X7F0, 0X7F1, 0X7F1, 0X7F3, 0X7F4, 0X7F6,
0X7F8, 0X7FA, 0X7FC, 0X7FD, 0X7FF, 0X7FE, 0X7FD, 0X7FC, 0X7FD, 0X7FF, 0X802, 0X806, 0X809, 0X80D, 0X80F, 0X811,
0X811, 0X811, 0X811, 0X810, 0X80F, 0X80D, 0X809, 0X807, 0X804, 0X801, 0X7FF, 0X7FC, 0X7F9, 0X7F5, 0X7F0, 0X7EB,
0X7E8, 0X7E7, 0X7E7, 0X7E8, 0X7E8, 0X7E9, 0X7E9, 0X7E9, 0X7E9, 0X7EB, 0X7ED, 0X7EF, 0X7F3, 0X7F8, 0X7FC, 0X801,
0X804, 0X806, 0X807, 0X807, 0X808, 0X80A, 0X80D, 0X810, 0X811, 0X811, 0X80F, 0X80C, 0X80A, 0X809, 0X808, 0X806,
0X805, 0X804, 0X803, 0X802, 0X7FF, 0X7FD, 0X7FD, 0X7FE, 0X7FF, 0X7FE, 0X7FC, 0X7FA, 0X7F9, 0X7F9, 0X7FB, 0X7FD,
0X7FE, 0X7FE, 0X7FE, 0X7FD, 0X7FC, 0X7FC, 0X7FD, 0X7FF, 0X800, 0X801, 0X802, 0X803, 0X806, 0X808, 0X80A, 0X80A,
0X808, 0X806, 0X807, 0X80A, 0X80D, 0X80F, 0X810, 0X80D, 0X809, 0X805, 0X804, 0X803, 0X803, 0X803, 0X802, 0X801,
```

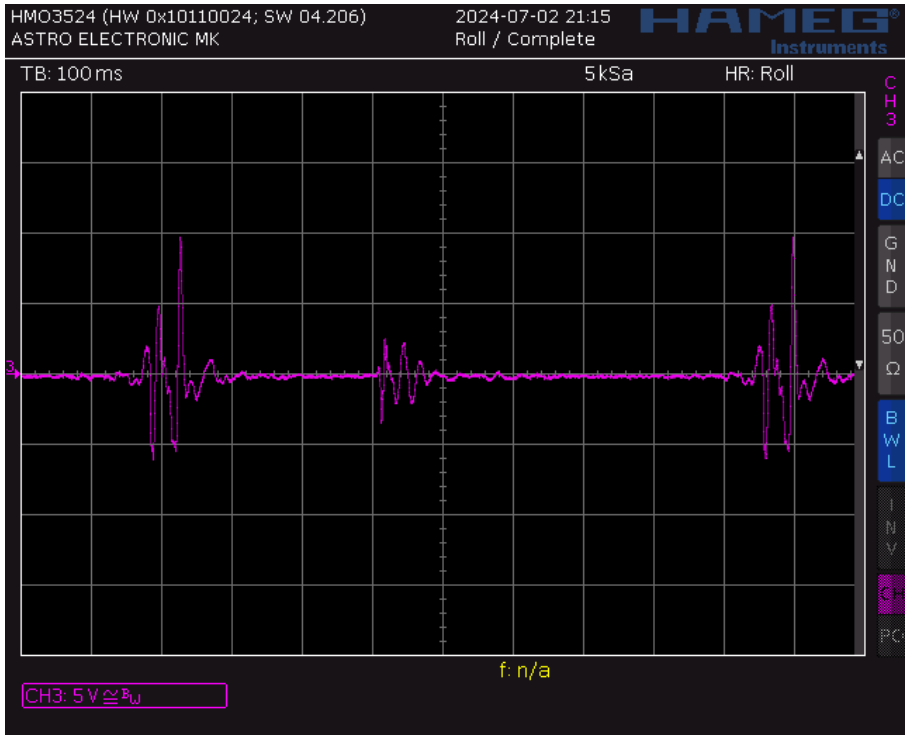
```
0X7FF, 0X7FD, 0X7FB, 0X7FB, 0X7FC, 0X7FE, 0X800, 0X801, 0X800, 0X7FF, 0X7FD, 0X7FB, 0X7FC, 0X7FE, 0X801, 0X802,
0X802, 0X801, 0X7FE, 0X7FB, 0X7F9, 0X7F7, 0X7F5, 0X7F5, 0X7F5, 0X7F4, 0X7F5, 0X7F4, 0X7F4, 0X7F4, 0X7F4, 0X7F5,
0X7F4, 0X7F5, 0X7F4, 0X7F2, 0X7EF, 0X7ED, 0X7EF, 0X7F5, 0X7FE, 0X805, 0X809, 0X808, 0X804, 0X803, 0X805, 0X809,
0X80E, 0X813, 0X814, 0X812, 0X810, 0X810, 0X810, 0X811, 0X813, 0X813, 0X813, 0X813, 0X813, 0X812, 0X812, 0X812,
0X813, 0X811, 0X80E, 0X80C, 0X80C, 0X80B, 0X80B, 0X808, 0X805, 0X802, 0X800, 0X801, 0X805, 0X808, 0X809, 0X80A,
0X80A, 0X809, 0X80A, 0X80D, 0X811, 0X816, 0X81F, 0X82F, 0X845, 0X85D, 0X87B, 0X891, 0X887, 0X857, 0X818, 0X7D3,
0X765, 0X6C3, 0X616, 0X58D, 0X555, 0X58F, 0X627, 0X6CB, 0X726, 0X728, 0X721, 0X772, 0X83B, 0X936, 0X9EF, 0XA0E,
0X9A1, 0X916, 0X8E2, 0X91F, 0X985, 0X9B5, 0X983, 0X90C, 0X898, 0X864, 0X87B, 0X8B2, 0X8CE, 0X8B5, 0X87E, 0X852,
0X844, 0X84F, 0X85B, 0X850, 0X82B, 0X7FC, 0X7D6, 0X7C7, 0X7CB, 0X7D0, 0X7C8, 0X7AE, 0X786, 0X75A, 0X72F, 0X708,
0X6E6, 0X6CA, 0X6AE, 0X694, 0X682, 0X67A, 0X67D, 0X68C, 0X6A2, 0X6B7, 0X6C9, 0X6DA, 0X6F4, 0X71A, 0X74B, 0X784,
0X7C1, 0X7FA, 0X82E, 0X85E, 0X891, 0X8CB, 0X908, 0X93F, 0X970, 0X999, 0X9B8, 0X9C7, 0X9CA, 0X9C7, 0X9CC, 0X9DB,
0X9E6, 0X9E0, 0X9C0, 0X98C, 0X94F, 0X917, 0X8EB, 0X8C7, 0X8A6, 0X881, 0X85C, 0X83A, 0X81D, 0X7FC, 0X7D0, 0X798,
0X75A, 0X722, 0X6FA, 0X6E0, 0X6CE, 0X6BC, 0X6A7, 0X694, 0X687, 0X683, 0X685, 0X689, 0X68F, 0X698, 0X6A5, 0X6B7,
0X6D0, 0X6ED, 0X70C, 0X72C, 0X74E, 0X770, 0X791, 0X7B2, 0X7D1, 0X7F0, 0X80F, 0X82C, 0X848, 0X866, 0X882, 0X89D,
0X8B5, 0X8C7, 0X8D4, 0X8DC, 0X8DD, 0X8DC, 0X8D9, 0X8D4, 0X8CC, 0X8C2, 0X8B6, 0X8A8, 0X897, 0X884, 0X870, 0X85B,
0X844, 0X82D, 0X815, 0X7FE, 0X7E8, 0X7D4, 0X7C3, 0X7B5, 0X7A9, 0X7A1, 0X79C, 0X799, 0X798, 0X79A, 0X79D, 0X7A2,
0X7A9, 0X7B0, 0X7B8, 0X7C2, 0X7CD, 0X7D9, 0X7E4, 0X7EF, 0X7F9, 0X803, 0X80C, 0X813, 0X81A, 0X820, 0X826, 0X82B,
0X830, 0X832, 0X833, 0X834, 0X834, 0X834, 0X836, 0X837, 0X837, 0X836, 0X833, 0X830, 0X82E, 0X82C, 0X82A, 0X828,
0X825, 0X823, 0X821, 0X81E, 0X81C, 0X81A, 0X818, 0X815, 0X812, 0X810, 0X80F, 0X80D, 0X80B, 0X80A, 0X80B, 0X80D,
0X80D, 0X80C, 0X809, 0X808, 0X805, 0X804, 0X801, 0X803, 0X800, 0X7F5, 0X7EC, 0X7EC, 0X7F2, 0X7F6, 0X7F8, 0X7F4,
0X7E8, 0X7DB, 0X7D3, 0X7D1, 0X7D4, 0X7D2, 0X7CE, 0X7CB, 0X7C2, 0X7BB, 0X7BD, 0X7C4, 0X7C8, 0X7CB, 0X7CC, 0X7CD,
0X7CD, 0X7CF, 0X7D3, 0X7D7, 0X7DA, 0X7DD, 0X7E2, 0X7E6, 0X7E9, 0X7EB, 0X7EC, 0X7EF, 0X7F3, 0X7FA, 0X801, 0X807,
0X80A, 0X80A, 0X80A, 0X80C, 0X810, 0X815, 0X819, 0X81C, 0X81D, 0X81B, 0X819, 0X817, 0X816, 0X816, 0X817, 0X816,
0X815, 0X814, 0X812, 0X810, 0X80F, 0X80E, 0X80F, 0X80F, 0X80F, 0X80F, 0X80F, 0X80D, 0X80B, 0X807, 0X803, 0X800,
0X7FF, 0X7FF, 0X7FE, 0X7FD, 0X7FB, 0X7FA, 0X7FA, 0X7FA, 0X7FA, 0X7F8, 0X7F7, 0X7F6, 0X7F4, 0X7F4, 0X7F6, 0X7F9,
0X7FB, 0X7FE, 0X802, 0X805, 0X804, 0X802, 0X802, 0X805, 0X80D, 0X814, 0X818, 0X819, 0X816, 0X811, 0X80E, 0X80D,
0X80E, 0X80F, 0X80F, 0X80E, 0X80C, 0X80B, 0X80B, 0X80C, 0X80D, 0X80D, 0X80C, 0X80B, 0X80C, 0X80F, 0X811, 0X812,
0X814, 0X815, 0X815, 0X813, 0X811, 0X810, 0X80E, 0X80B, 0X808, 0X807, 0X804, 0X800, 0X7FD, 0X7FC, 0X7FC, 0X7FA,
0X7F7, 0X7F3, 0X7F0, 0X7ED, 0X7EB, 0X7EA, 0X7E7, 0X7E5, 0X7E2, 0X7E0, 0X7DE, 0X7DA, 0X7D6, 0X7D3, 0X7D0, 0X7D0,
0X7D1, 0X7D3, 0X7D7, 0X7DA, 0X7DC, 0X7DF, 0X7E2, 0X7E6, 0X7EB, 0X7F0, 0X7F6, 0X7FA, 0X7FF, 0X802, 0X805, 0X808,
0X80B, 0X810, 0X818, 0X821, 0X82B, 0X833, 0X838, 0X83A, 0X838, 0X836, 0X836, 0X838, 0X838, 0X837, 0X833, 0X82E,
```

```
0X828, 0X823, 0X81E, 0X819, 0X814, 0X80F, 0X80B, 0X807, 0X803, 0X7FF, 0X7FC, 0X7F9, 0X7F6, 0X7F3, 0X7F0, 0X7ED,
0X7EB, 0X7EA, 0X7EA, 0X7EA, 0X7EB, 0X7E9, 0X7E6, 0X7E4, 0X7E2, 0X7E1, 0X7E3, 0X7E5, 0X7E7, 0X7E9, 0X7E9, 0X7E7,
0X7E4, 0X7E0, 0X7E0, 0X7E3, 0X7E7, 0X7EB, 0X7EB, 0X7E8, 0X7E4, 0X7E1, 0X7E0, 0X7E4, 0X7E8, 0X7EC, 0X7EE, 0X7EF,
0X7EE, 0X7ED, 0X7EC, 0X7ED, 0X7EE, 0X7F0, 0X7F2, 0X7F4, 0X7F6, 0X7F8, 0X7FB, 0X7FD, 0X800, 0X802, 0X804, 0X807,
0X80B, 0X811, 0X815, 0X819, 0X81A, 0X819, 0X819, 0X81B, 0X81D, 0X81E, 0X81C, 0X819, 0X816, 0X814, 0X813, 0X812,
0X810, 0X80F, 0X80D, 0X80C, 0X80C, 0X80D, 0X80B, 0X807, 0X7FF, 0X7F8, 0X7F2, 0X7F0, 0X7F1, 0X7F3, 0X7F3, 0X7F1,
0X7EF, 0X7ED, 0X7ED, 0X7EF, 0X7F5, 0X7FD, 0X803, 0X80A, 0X80E, 0X812, 0X815, 0X817, 0X817, 0X817, 0X815, 0X813,
0X811, 0X811, 0X813, 0X815, 0X814, 0X812, 0X80F, 0X80E, 0X811, 0X814, 0X817, 0X817, 0X812, 0X80C, 0X807, 0X806,
0X806, 0X807, 0X80A, 0X80D, 0X80E, 0X80A, 0X804, 0X7FD, 0X7FA, 0X7FC, 0X800, 0X803, 0X803, 0X800, 0X7FA, 0X7F7,
0X7F6, 0X7F6, 0X7F6, 0X7F5, 0X7F4, 0X7F4, 0X7F5, 0X7F7, 0X7F9, 0X7F8, 0X7F6, 0X7F3, 0X7F2, 0X7F2, 0X7F2, 0X7F3,
0X7F3, 0X7F5, 0X7F9, 0X7FD, 0X7FD, 0X7F7, 0X7EC, 0X7E1, 0X7DC, 0X7DF, 0X7E7, 0X7F0, 0X7F8, 0X7FA, 0X7F7, 0X7F4,
0X7F1, 0X7F3, 0X7F6, 0X7F8, 0X7F7, 0X7F6, 0X7F3, 0X7F1, 0X7F3, 0X7F7, 0X7FB, 0X7FD, 0X7FC, 0X7F9, 0X7F6, 0X7F5,
0X7F5, 0X7F8, 0X7FB, 0X800, 0X804, 0X808, 0X80B, 0X80F, 0X810, 0X811, 0X813, 0X813, 0X813, 0X812, 0X812, 0X811,
0X810, 0X810, 0X80F, 0X80E, 0X80B, 0X808, 0X804, 0X801, 0X7FF, 0X7FD, 0X7FD, 0X7FC, 0X7FC, 0X7FB, 0X7FA, 0X7F9,
0X7F9, 0X7F8, 0X7F8, 0X7F9, 0X7F9, 0X7F8, 0X7F7, 0X7F6, 0X7F8, 0X7FA, 0X7FD, 0X7FE, 0X7FD, 0X7FB, 0X7FA, 0X7FA,
0X7FC, 0X7FF, 0X803, 0X806, 0X807, 0X805, 0X803, 0X802, 0X802, 0X801, 0X800, 0X7FF, 0X801, 0X803, 0X806, 0X807,
0X807, 0X804, 0X802, 0X7FF, 0X7FC, 0X7FA, 0X7FA, 0X7FA, 0X7FB, 0X7FD, 0X7FE, 0X7FF, 0X801, 0X803, 0X804, 0X804,
0X803, 0X803, 0X803, 0X804, 0X806, 0X809, 0X80C, 0X80E, 0X80F, 0X810, 0X80F, 0X80F, 0X80E, 0X80D, 0X80B, 0X808,
0X805, 0X802, 0X800, 0X7FF, 0X7FD, 0X7FB, 0X7FA, 0X7F9, 0X7F8, 0X7F6, 0X7F5, 0X7F5, 0X7F5, 0X7F5, 0X7F5, 0X7F6,
0X7F7, 0X7F8, 0X7F9, 0X7F9, 0X7FA, 0X7FB, 0X7FB, 0X7FC, 0X7FC, 0X7FC, 0X7FB, 0X7FA, 0X7FB, 0X7FD, 0X7FD, 0X7FE,
0X7FD, 0X7FD, 0X7FE, 0X7FF, 0X802, 0X805, 0X805, 0X804, 0X801, 0X7FE, 0X7FC, 0X7FC, 0X7FF, 0X804, 0X808, 0X80B,
0X80A, 0X808, 0X804, 0X802, 0X803, 0X803, 0X804, 0X804, 0X803, 0X802, 0X800, 0X7FE, 0X7FD, 0X7FD, 0X7FC, 0X7FC,
0X7FD, 0X7FE, 0X7FF, 0X7FF, 0X800, 0X800, 0X800, 0X801, 0X801, 0X801, 0X800, 0X7FF, 0X7FE, 0X7FE, 0X7FD, 0X7FB,
0X7FB, 0X7FA, 0X7F9, 0X7F9, 0X7F8, 0X7F6, 0X7F5, 0X7F4, 0X7F3, 0X7F4, 0X7F5, 0X7F6, 0X7F6, 0X7F7, 0X7F6, 0X7F5,
0X7F5, 0X7F7, 0X7FB, 0X7FE, 0X800, 0X801, 0X800, 0X7FE, 0X7FF, 0X801, 0X803, 0X805, 0X806, 0X806, 0X806, 0X806,
0X807, 0X809, 0X80B, 0X80C, 0X80E, 0X80D, 0X80D, 0X80C, 0X80C, 0X80D, 0X80F, 0X80E, 0X80D, 0X809, 0X805, 0X802,
0X7FF, 0X7FE, 0X7FD, 0X7FD, 0X7FE, 0X7FE, 0X7FD, 0X7FB, 0X7FB, 0X7FB, 0X7FB, 0X7FD, 0X7FF, 0X7FF, 0X7FD, 0X7FB,
0X7F9, 0X7F6, 0X7F4, 0X7F4, 0X7F7, 0X7FC, 0X800, 0X804, 0X805, 0X804, 0X803, 0X800, 0X7FE, 0X7FF, 0X801, 0X805,
0X807, 0X807, 0X805, 0X801, 0X7FD, 0X7FA, 0X7F8, 0X7F7, 0X7F7, 0X7F8, 0X7F8, 0X7FB, 0X7FD, 0X801, 0X804, 0X807,
0X808, 0X809, 0X809, 0X809, 0X808, 0X808, 0X807, 0X806, 0X804, 0X802, 0X7FF, 0X7FB, 0X7F7, 0X7F2, 0X7F0, 0X7F2,
0X7F5, 0X7F7, 0X7F8, 0X7F8, 0X7F7, 0X7F5, 0X7F6, 0X7F8, 0X7FC, 0X800, 0X803, 0X805, 0X805, 0X803, 0X7FF, 0X7FC,
```

```
 0X7FB, 0X7FB, 0X7FA, 0X7FA, 0X7FA, 0X7F9, 0X7F9, 0X7FB, 0X7FD, 0X7FF, 0X7FE, 0X7FF, 0X800, 0X803, 0X803, 0X803,
 0X804, 0X804, 0X805, 0X805, 0X805, 0X806, 0X807, 0X807, 0X806, 0X804, 0X802, 0X801, 0X803, 0X804, 0X803, 0X801,
 0X7FF, 0X7FF, 0X800, 0X803, 0X806, 0X807, 0X807, 0X806, 0X806, 0X806, 0X807, 0X807, 0X807, 0X806, 0X804, 0X803,
 0X802, 0X800, 0X7FD, 0X7FA, 0X7F8, 0X7F5, 0X7F2, 0X7F1, 0X7F1, 0X7F1, 0X7F3, 0X7F5, 0X7F6, 0X7F5, 0X7F4, 0X7F3,
 0X7F4, 0X7F6, 0X7F8, 0X7FA, 0X7FD, 0X800, 0X802, 0X803, 0X802, 0X801, 0X7FE, 0X7F8, 0X7F1, 0X7EF, 0X7F1, 0X7F7,
 0X7FF, 0X807, 0X80C, 0X80B, 0X808, 0X806, 0X804, 0X804, 0X805, 0X805, 0X806, 0X807, 0X808, 0X80B, 0X80E, 0X811,
 0X811, 0X811, 0X80F, 0X80D, 0X80D, 0X80D, 0X80E, 0X80F, 0X80D, 0X80B, 0X809, 0X807, 0X804, 0X803, 0X803, 0X802,
 0X800, 0X7FF, 0X800, 0X802, 0X806, 0X806, 0X803, 0X7FE, 0X7F7, 0X7F2, 0X7F2, 0X7F5, 0X7FB, 0X7FF, 0X801, 0X7FF,
 0X7FA, 0X7F4, 0X7EF, 0X7EA, 0X7E6, 0X7E3, 0X7E1, 0X7DF, 0X7DF, 0X7E0, 0X7E2, 0X7E6, 0X7E8, 0X7EA, 0X7ED, 0X7EF,
 0X7F4, 0X7FC, 0X804, 0X80A, 0X80F, 0X812, 0X813, 0X814, 0X815, 0X816, 0X817, 0X818, 0X818, 0X818, 0X816, 0X814,
 0X813, 0X811, 0X810, 0X80E, 0X80B, 0X808, 0X804, 0X7FF, 0X7FA, 0X7F7, 0X7F7, 0X7F9, 0X7FB, 0X7FC, 0X7FC, 0X7F8,
 0X7F4, 0X7EF, 0X7EC, 0X7EA, 0X7E8, 0X7E7, 0X7E7, 0X7E7, 0X7E9, 0X7ED, 0X7EF, 0X7F2, 0X7F5, 0X7FA, 0X800, 0X805,
 0X808, 0X809, 0X807, 0X805, 0X803, 0X804, 0X804, 0X806, 0X808, 0X80A, 0X80B, 0X80B, 0X80B, 0X80C, 0X80C, 0X80D,
 0X80D, 0X80D, 0X80D, 0X80D, 0X80C, 0X80C, 0X80C, 0X80C, 0X80C, 0X80B, 0X80B, 0X80A, 0X809, 0X807, 0X805, 0X803
};

unsigned int ph = 0;
unsigned int dp = 0;
IntervalTimer myTimer1;
IntervalTimer myTimer2;

void setup()    // the setup routine runs once when you press reset
{
  pinMode(led, OUTPUT);        // initialize the digital pin as an output
  analogReference(EXTERNAL);   // 3.0V reference
  analogReadResolution(12);    // set ADC to 12 bits
  analogWriteResolution(12);   // set DAC to 12 bits

  myTimer1.begin(update_frequency, 1000);  // set the frequency update interval to 1000 microseconds (1kHz)
  myTimer2.begin(timer_interrupt, 8);  // set the sample interval to 8 microseconds (125kHz)
  myTimer2.priority(0);                // high priority for timer interrupt
```

```
}

void update_frequency()
{
  digitalWrite(led, HIGH);                  // switch on the LED
  dp = analogRead(A0) * 0.139844 * freq;    // calculate the value that is added to the 32-bit phase accumulator
                                            //  2^32 / 125000 / 4095 / 60 = 0.139844
  digitalWrite(led, LOW);                   // switch off the LED
}

void timer_interrupt()
{
  ph += dp;                                 // update the phase accumulator
  analogWrite(A12, waveform[ph >> 21]);     // use the most significant 11 bits as index for waveform array, and write to
DAC
}

void loop()   // nothing to do in the main loop
{
}
```

Output signal:



4.4 Ideas for Hybrid Computer with Teensy 4.0

Wishes:

- **Keep it simple**
- **Digital potentiometers are typically set before the computer is started, and not changed while the computer is running. Most users will be happy with statically set potentiometers (but I'm not)**
- **Stand-alone usage is important for many users. Especially for schools. Best if it could be used without computer and without oscilloscope.**
  Problem: Stand-alone means a formula parser must be implemented. That's not easy, and it's slower than having the user-defined functions as compiled C code.
  The alternative is to use the Arduino/Teensy IDE for implementing the user-defined functions. The source code can be split in two files: One file for user-defined things, and the other file for the operating system.
- **There are good color OLED displays with touch interface**
- **Additionally the hybrid part can be controlled over USB interface from a PC, using a simple command language. For example for setting of the digital potentiometers.**
- **AD5293 has 10-bit values, set via SPI. Conversion from [0..1] to [0..1023]**
- **+-1 Machine unit = +-10V --> 12-bit 4.9mV, 14-bit 1.2mV, 16-bit 0.3mV --> 14-bit is sufficient!**

Ideas how the digital potentiometers (or DACs) can be controlled:

- **Statically set a value from the PC via USB interface, using a simple command language.**
- **Function generator with one input: out=f(x) where x is a dynamic input signal from a ADC, realised with look-up-table, updated with constant sample rate.**
- **Function generator with multiple inputs: out=f(x,y,...) where x,y,... are dynamic input signals from ADCs, calculation in realtime, a constant sample rate can't be guaranteed.**
- **Signal generator with constant frequency: out=f(omega · t) with selectable frequency and waveform (sine, square, triangle, sawtooth, ...) and selectable phase shift**
- **Signal generator with variable frequency: out=f(x · omega · t) where x is a dynamic input signal from a ADC, with selectable frequency and waveform (sine, square, triangle, sawtooth, ...) and selectable phase shift. No phase jump when the frequency is changed.**

Which pins from THATs hybrid port should be used?

Pin	Name	Function	Use it?
2	HYB-X	low impedance analog output signals, voltage is 0.1 · OUT-X + 1.64V, range 0.64V to 2.64V for +-1 MU	These output signals can be used for an oscilloscope.
4	HYB-Y		
6	HYB-Z		
8	HYB-U		
1	IN-X	The purpose of these signals is unclear.	no
3	IN-Y		
5	IN-Z		
7	IN-U		
9, 10	GND	ground	yes
11, 12	VUSB	is connected to +5V USB supply voltage via 750mA PTC fuse	no, the hybrid part should have its own USB supply.
13	DIR	This is an input with 3.3V level, low level enables THATs hybrid mode. The input has an internal pullup resistor to 3.3V and it not 5V tolerant.	yes
14	-Mode_OP	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = operate In hybrid mode: 5V tolerant 3.3V input, sets THATs operation mode, the input has no internal pullup resistor	yes
15	Voffset	+1.64V	it depends
16	-Mode_IC	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = initial condition In hybrid mode: 5V tolerant 3.3V input, sets THATs initial condition mode, the input has no internal pullup resistor	yes

Note: Looking from the rear side into the hybrid port: Pin 1 is top right, Pin 2 is bottom right.

How the hybrid port works:

DIR must be pulled low to set THAT in "Hybrid" mode. The position of the MODE switch doesn't care.

Low level at -ModeOP sets THAT in "Operate" mode.

Low level at -ModeIC sets THAT in "Initial Condition" mode.

4.5 Seeed XIAO SAMD21

More info about Seeed XIAO SAMD21:

https://wiki.seeedstudio.com/SeeedStudio_XIAO_Series_Introduction/

https://wiki.seeedstudio.com/Seeeduino-XIAO/

SAMD21 Data sheet: https://files.seeedstudio.com/wiki/Seeeduino-XIAO/res/ATSAMD21G18A-MU-Datasheet.pdf

Schematic diagram: https://files.seeedstudio.com/wiki/Seeeduino-XIAO/res/Seeeduino-XIAO-v1.0-SCH-191112.pdf

Wearable Projects Step by Step: https://files.seeedstudio.com/wiki/Seeeduino-XIAO/res/Seeeduino-XIAO-in-Action-Minitype%EF%BC%86Wearable-Projects-Step-by-Step.pdf

The ADC is 12-bit and the DAC is 10-bit.

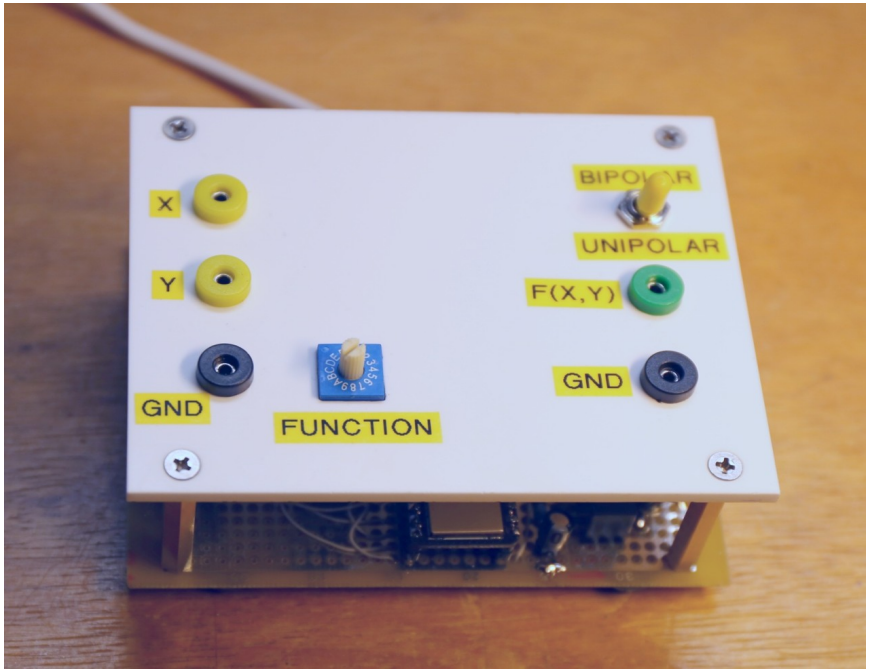
The clock frequency is 48 MHz, derived via PLL from a 32.768 kHz crystal.

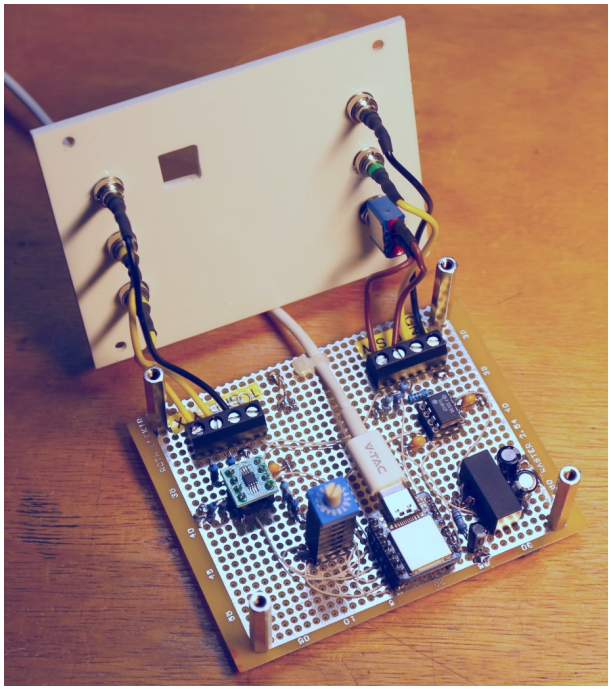
4.5.1 Function Generator

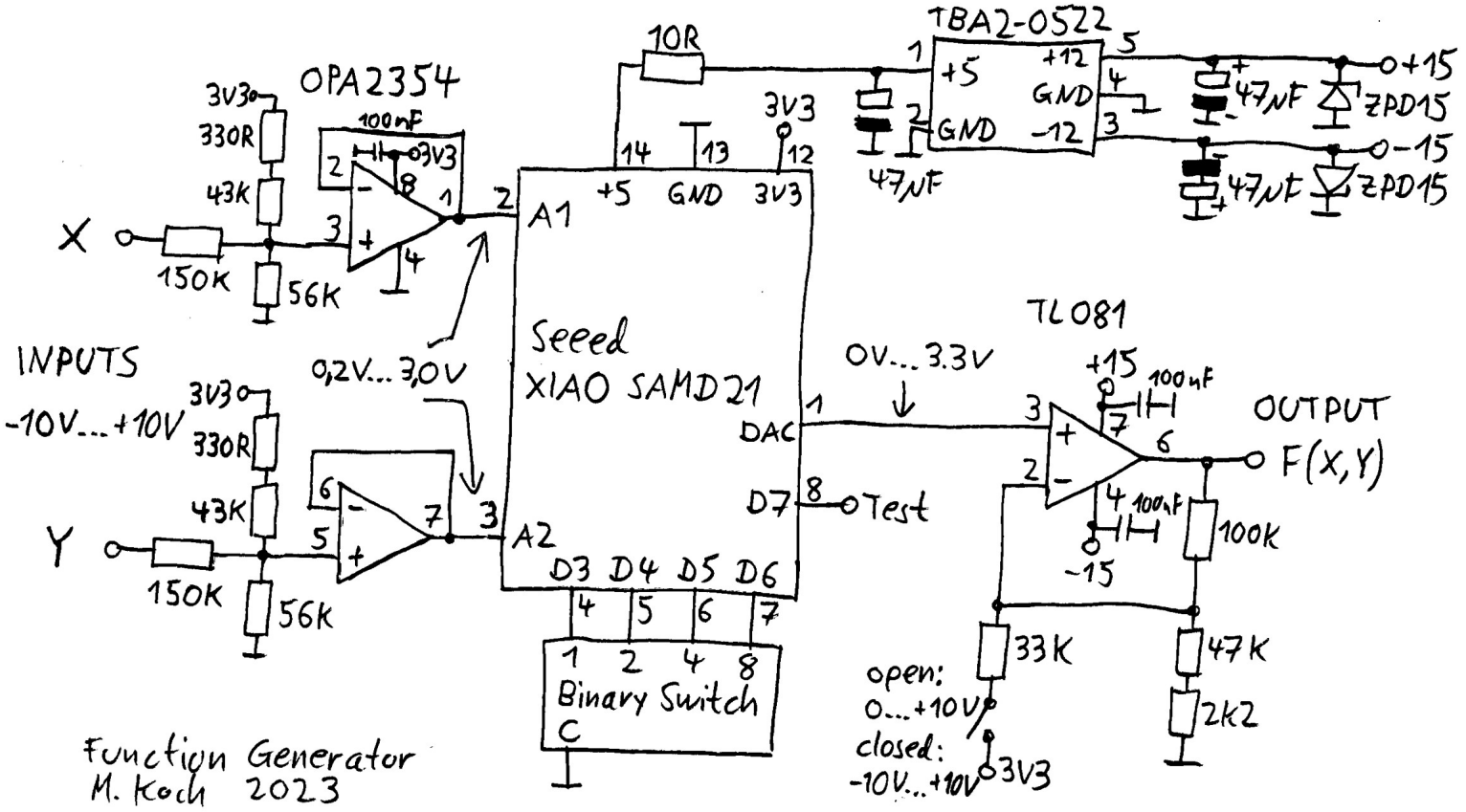
This is a small function generator with the Seeed XIAO SAMD21 microcontroller. It has two analog -10V to +10V inputs (+-1 machine unit) and one analog output which can be switched to unipolar mode (0 to +10V) or bipolar mode (-10V to +10V). 16 different functions F(X,Y) can be selected with a binary switch. The software doesn't use lookup tables. The functions are calculated in realtime. The samplerate is between 3 and 10 kHz. I did try to make the circuit as simple as possible. There are no potentiometers for offset and input gain. That can be adjusted in software.

List of functions:
x·y, x^2, x^3, sqrt(x), x^(1/3), sgn(x), x/(10y), sqrt(x·x+y·y), x^(5y), exp(x-1), exp(2x-2), 1+ln(x), 1+0.5·ln(x), atan2(y/x)/pi, sin(x·pi), cos(x·pi)

But of course you can change the functions in software.







Although the OPA2354 has rail-to-rail inputs and outputs, it wasn't able to drive the analog input of the microcontroller close enough to the 3.3V rail. That's why I'm using the voltage range from 0.2V to 3.0V. The ADC has 12-bit resolution and the input range is from 0 to 3.3V. That means I'm using here not the full 12 bits. The effective resolution is $\log_2(4096 \cdot 2.8V / 3.3V) = 11.7$ bits.
The DAC has only 10-bit resolution and the full 0 to 3.3V range is used. The resolution is about 20mV in bipolar mode or 10mV in unipolar mode.

Note: The 10R resistor at the input of the voltage converter was inserted to make sure that the current can never become larger than 0.5 A. This resistor is unnecessary because the voltage converter has an internal overload protection.
With shorted +-12V outputs, the input current is less than 0.2 A. With worst case load (about 56 Ohms at each output), the input current is less than 0.8 A.

```
/*
  Programmable Function Generator for THAT with Seeed_XIAO
  Michael Koch, December 2023
*/

bool bipolar;
int mode;
float x,y,f;


// the setup function runs once when you press reset or power the board
void setup() {
  //pinMode(LED_BUILTIN, OUTPUT);
  pinMode(A1,INPUT);           // analog inputs
  pinMode(A2,INPUT);
  pinMode(D3, INPUT_PULLUP);   // pullup resistors for mode switch inputs
  pinMode(D4, INPUT_PULLUP);
  pinMode(D5, INPUT_PULLUP);
  pinMode(D6, INPUT_PULLUP);
  pinMode(D7, OUTPUT);         // Test pin for sample rate
  analogReadResolution(12);
  analogWriteResolution(10);
}

// the loop function runs over and over again forever
void loop() {
  mode = 15 - digitalRead(D3);  // read the mode switch
  mode -= 2 * digitalRead(D4);
  mode -= 4 * digitalRead(D5);
  mode -= 8 * digitalRead(D6);

  // You can fine-tune the offset and gain values!
  digitalWrite(D7, HIGH);                // test pin = high
  x = -1.15 + 0.00058 * analogRead(A1);  // read analog x input
  digitalWrite(D7, LOW);                 // test pin = low
  y = -1.15 + 0.00058 * analogRead(A2);  // read analog y input

  switch (mode)
  {
    case 0:  f = x * y;
             bipolar = true;
             break;
```

```
    case 1:  f = x * x;
             bipolar = false;
             break;
    case 2:  f = x * x * x;
             bipolar = true;
             break;
    case 3:  f = (x>0) ? sqrt(x) : 0;
             bipolar = false;
             break;
    case 4:  f = (x>0) ? pow(x,0.3333333) : 0;
             bipolar = false;
             break;
    case 5:  f = (x>0) ? 1 : -1;
             bipolar = true;
             break;
    case 6:  f = x / (10 * y);
             bipolar = true;
             break;
    case 7:  f = sqrt(x*x+y*y);
             bipolar = false;
             break;
    case 8:  f = pow(x,5*y);
             bipolar = false;
             break;
    case 9:  f = exp(x-1);
             bipolar = false;
             break;
    case 10: f = exp(2*x - 2);
             bipolar = false;
             break;
    case 11: f = (x>0) ? 1 + log(x) : -1;
             bipolar = true;
             break;
    case 12: f = (x>0) ? 1 + 0.5 * log(x) : -1;
             bipolar = true;
             break;
    case 13: f = atan(y/x) / 3.141592; // -0.5 ... +0.5
             if (x<0)
               if (y>0) f += 1;        // +0.5 ... +1.0
               else f -= 1;            // -1.0 ... -0.5
             bipolar = true;
             break;
```

```
    case 14: f = sin(x * 3.141592);
             bipolar = true;
             break;
    case 15: f = cos(x * 3.141592);
             bipolar = true;
             break;
  }

  if (bipolar)
  {
    if (f < -1) f = -1;
    if (f > 1) f = 1;
    analogWrite(A0,(int)(511.5 + f * 511.5));   // bipolar output
  }
  else
  {
    if (f < 0) f = 0;
    if (f > 1) f = 1;
    analogWrite(A0,(int)(f * 1023));            // unipolar output
  }
}
```

This is an improved version of the software, using look-up-tables for all functions that depend only on one variable. These functions are now executed with about 50 kHz samplerate. Functions of two variables are slower:
x·y 29 kHz
x/(10y) 14.6 kHz
sqrt(x·x+y·y) 18.3 kHz
x^(5y) 4 kHz
atan2(y/x)/3.141592 4.7 kHz

```
/*
  Programmable Function Generator for THAT with Seeed_XIAO
  Version 2 with look-up-tables
  Michael Koch, January 2024
*/

bool bipolar;
int mode, lastMode;
unsigned int port;
float x,y,f;
int xx, yy;
float offset, gain;
union
{
  int i[4096];
  float f[4096];
} lut;

// the setup function runs once when you press reset or power the board
void setup()
{
  offset = -1.15;      // This can be fine-tuned
  gain = 0.00058;      // This can be fine-tuned
  lastMode = 255;
  pinMode(A1,INPUT);           // analog inputs
  pinMode(A2,INPUT);
  pinMode(D3, INPUT_PULLUP);   // pullup resistors for mode switch inputs
  pinMode(D4, INPUT_PULLUP);
  pinMode(D5, INPUT_PULLUP);
  pinMode(D6, INPUT_PULLUP);
  pinMode(D7, OUTPUT);         // Test pin for sample rate
  analogReadResolution(12);
  analogWriteResolution(10);
}
```

```
void fillLut(int mode)
{
  int i;
  float f;
  for (int x = 0; x < 4096; x++)
  {
    switch (mode)
    {
      case 0: f = offset + gain * x;
              if (f < -1) f = -1;
              if (f > 1) f = 1;
              lut.i[x] = (int)(2048 * f);
              break;
      case 1: i = (int)(1023 * sq(offset + gain * x));
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 2: i = (int)(511.5 + 511.5 * pow(offset + gain * x, 3));
              if (i < 0) i = 0;
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 3: f = offset + gain * x;
              f = (f>0) ? sqrt(f) : 0;
              i = (int)(1023 * f);
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 4: f = offset + gain * x;
              f = (f>0) ? pow(f, 0.33333) : 0;
              i = (int)(1023 * f);
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 5: break;
      case 6: f = offset + gain * x;
              if (f < -1) f = -1;
              if (f > 1) f = 1;
              lut.f[x] = f;
              break;
      case 7: f = sq(offset + gain * x);
              if (f > 1) f = 1;
              lut.f[x] = f;
```

```
            break;
    case 8: f = offset + gain * x;
            if (f < -1) f = -1;
            if (f > 1) f = 1;
            lut.f[x] = f;
            break;
    case 9: f = offset + gain * x;
            f = exp(f - 1.0);
            i = (int)(1023 * f);
            if (i > 1023) i = 1023;
            lut.i[x] = i;
            break;
    case 10: f = offset + gain * x;
             f = exp(2 * f - 2.0);
             i = (int)(1023 * f);
             if (i > 1023) i = 1023;
             lut.i[x] = i;
             break;
    case 11: f = offset + gain * x;
             f = (f > -1) ? log(1 + f) : -1;
             if (f < -1) f = -1;
             if (f>1) f = 1;
             i = (int)(511.5 + 511.5 * f);
             lut.i[x] = i;
             break;
    case 12: f = offset + gain * x;
             f = (f > -1) ? 0.5 * log(1 + f) : -1;
             if (f < -1) f = -1;
             if (f>1) f = 1;
             i = (int)(511.5 + 511.5 * f);
             lut.i[x] = i;
             break;
    case 13: f = offset + gain * x;
             if (f < -1) f = -1;
             if (f > 1) f = 1;
             lut.f[x] = f;
             break;
    case 14: f = offset + gain * x;
             if (f < -1) f = -1;
             if (f > 1) f = 1;
             f = sin(f * 3.141592);
             lut.i[x] = (int)(511.5 + 511.5 * f);
             break;
```

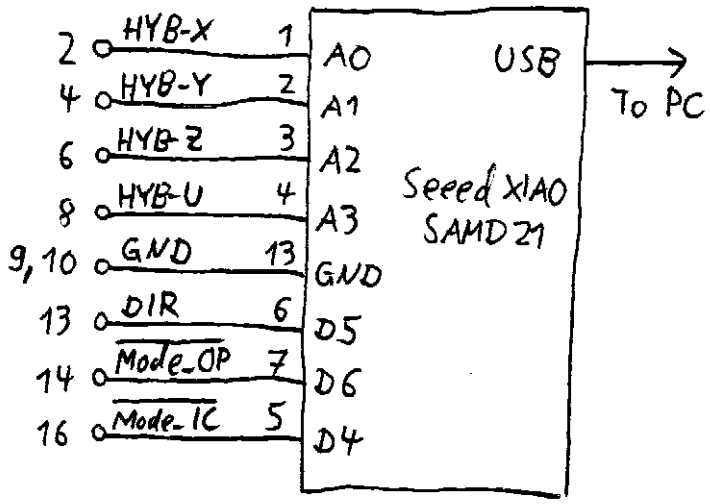
```
      case 15: f = offset + gain * x;
               if (f < -1) f = -1;
               if (f > 1) f = 1;
               f = cos(f * 3.141592);
               lut.i[x] = (int)(511.5 + 511.5 * f);
               break;
    }
  }
}

// the loop function runs over and over again forever
void loop()
{
  // mode = 15 - digitalRead(D3);  // read the mode switch, this is slow
  // mode -= 2 * digitalRead(D4);
  // mode -= 4 * digitalRead(D5);
  // mode -= 8 * digitalRead(D6);
  mode = 15;
  port = PORT->Group[0].IN.reg;    // read the mode switch, this is faster
  if (port & 2048) mode -= 1;
  if (port & 256) mode -= 2;
  if (port & 512) mode -= 4;
  port = PORT->Group[1].IN.reg;
  if (port & 256) mode -= 8;

  if (mode != lastMode)         // if the mode has changed,
  {
    fillLut(mode);              // fill the look-up-table
    lastMode = mode;
  }

  // digitalWrite(D7, HIGH);                // test pin = high, this is slow
  PORT->Group[1].OUTSET.reg = 512;          // test pin = high, this is faster
  xx = analogRead(A1);    // read analog x input
  // digitalWrite(D7, LOW);                 // test pin = low, this is slow
  PORT->Group[1].OUTCLR.reg = 512;          // test pin = low, this is faster

  switch (mode)
  {
    case 0:  yy = analogRead(A2);        // x * y
             analogWrite(A0,511 + 512 * lut.i[xx] * lut.i[yy] / 4194304);
             break;
    case 1:  analogWrite(A0,lut.i[xx]);  // x^2
```

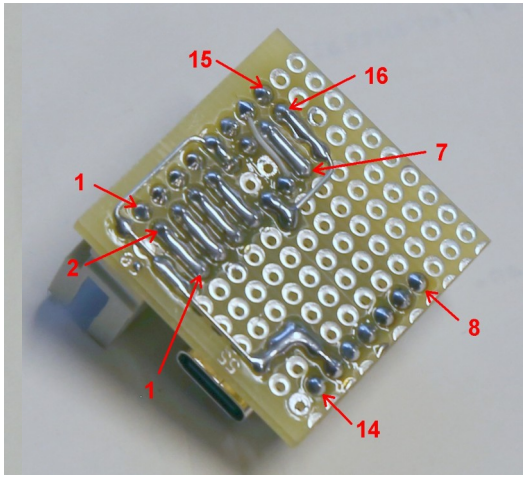
```
          break;
  case 2: analogWrite(A0,lut.i[xx]);  // x^3
          break;
  case 3: analogWrite(A0,lut.i[xx]);  // sqrt(x)
          break;
  case 4: analogWrite(A0,lut.i[xx]);  // x^(1/3)
          break;
  case 5: if (xx < 2048)              // sgn(x)
            analogWrite(A0,0);
          else
            analogWrite(A0,1023);
          break;
  case 6: yy = analogRead(A2);        // x / (10 * y)
          f = lut.f[xx] / (10 * lut.f[yy]);
          if (f < -1) f = -1;
          if (f > 1) f = 1;
          analogWrite(A0,(int)(511.5 + f * 511.5));
          break;
  case 7: yy = analogRead(A2);        // sqrt(x*x+y*y)
          f = sqrt(lut.f[xx] + lut.f[yy]);
          if (f > 1) f = 1;
          analogWrite(A0,(int)(f * 1023));
          break;
  case 8: yy = analogRead(A2);        // x^(5*y)
          f = (xx > 2047) ? pow(lut.f[xx], 5 * lut.f[yy]) : 0;
          if (f > 1) f = 1;
          analogWrite(A0,(int)(f * 1023));
          break;
  case 9: analogWrite(A0,lut.i[xx]);  // exp(x-1)
          break;
  case 10: analogWrite(A0,lut.i[xx]);  // exp(2x-2)
          break;
  case 11: analogWrite(A0,lut.i[xx]);  // log(1+x)
          break;
  case 12: analogWrite(A0,lut.i[xx]);  // 0.5 * log(1+x)
          break;
  case 13: yy = analogRead(A2);        // atan2(y/x)/3.141592
          x = lut.f[xx];
          y = lut.f[yy];
          f = atan(y/x) / 3.141592; // -0.5 ... +0.5
          if (x<0)
            if (y>0) f += 1;        // +0.5 ... +1.0
            else f -= 1;            // -1.0 ... -0.5
```

```
            if (f < -1) f = -1;
            if (f > 1) f = 1;
            analogWrite(A0,(int)(511.5 + f * 511.5));
            break;
    case 14: analogWrite(A0,lut.i[xx]);  // sin(x * 3.141592)
            break;
    case 15: analogWrite(A0,lut.i[xx]);  // cos(x * 3.141592)
            break;
  }
}
```

4.5.2 Hybrid Port Oscilloscope

The schematic diagram is very simple. If you need a 4-channel oscilloscope for THAT which can be built for less than 30 EUR, here it is!
The pin numbers at the left side are for THATs hybrid port:





It's important that pins 11 and 12 of THATs hybrid port aren't connected to any pins of the microcontroller, because these are +5V pins and the microcontroller's input and output pins aren't 5V tolerant. The microcontroller gets its own supply voltage over the USB cable.

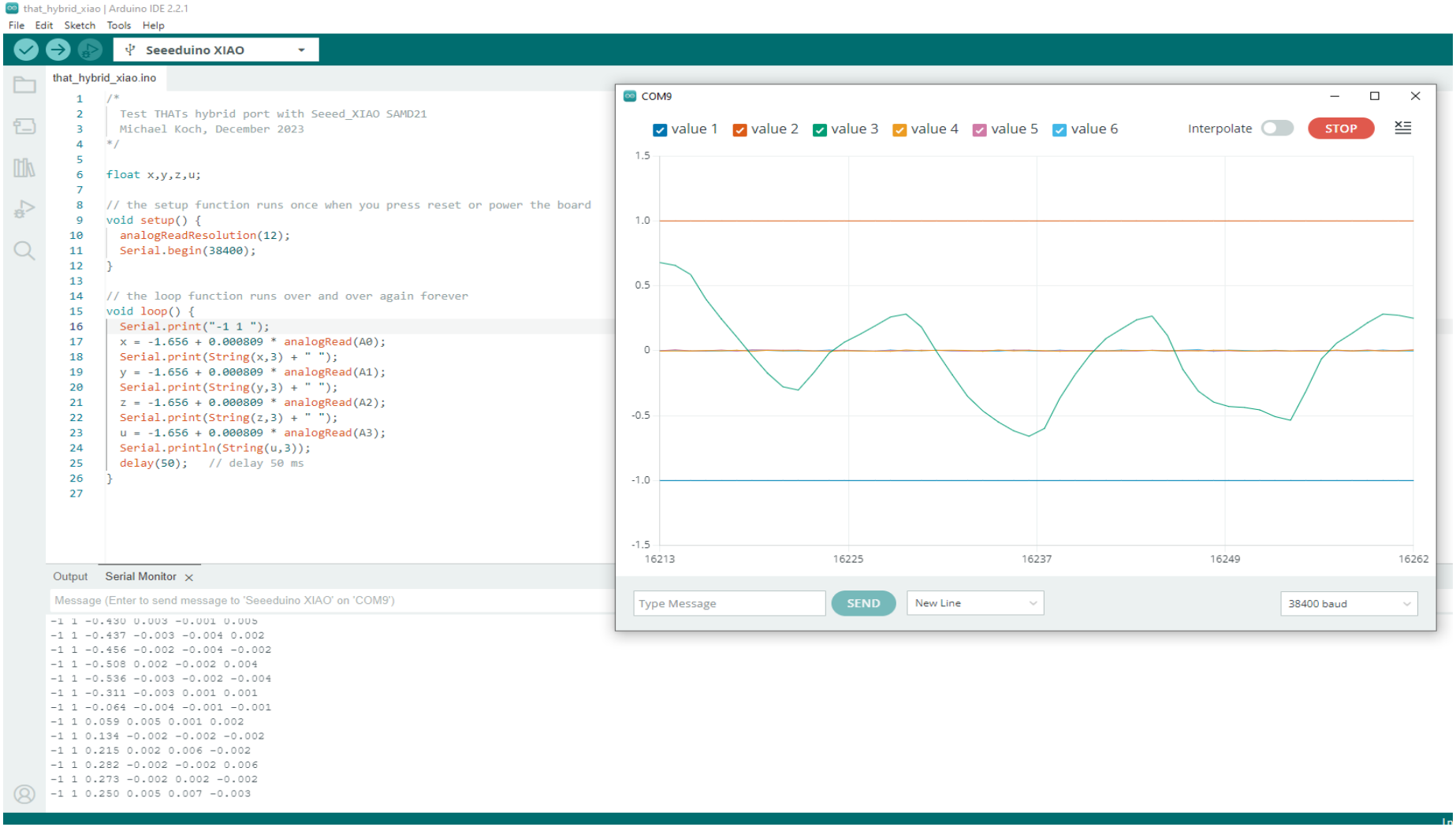
This is a very simple (preliminary) software for testing the hybrid port oscilloscope. It's much easier than I thought at first, because the Arduino IDE has already a built-in serial plotter. The software just reads four analog inputs, converts the values to machine units and writes the results to the serial port. The sampling speed can be changed in the last line of the source code.

```
/*
  Test THATs hybrid port with Seeed_XIAO SAMD21
  Michael Koch, December 2023
*/

float x,y,z,u;

// the setup function runs once when you press reset or power the board
void setup() {
  analogReadResolution(12);
}

// the loop function runs over and over again forever
void loop() {
  Serial.print("-1 1 ");
  x = -1.656 + 0.000809 * analogRead(A0);
  Serial.print(String(x,3) + " ");
  y = -1.656 + 0.000809 * analogRead(A1);
  Serial.print(String(y,3) + " ");
  z = -1.656 + 0.000809 * analogRead(A2);
  Serial.print(String(z,3) + " ");
  u = -1.656 + 0.000809 * analogRead(A3);
  Serial.println(String(u,3));
  delay(50);   // delay 50 ms
}
```

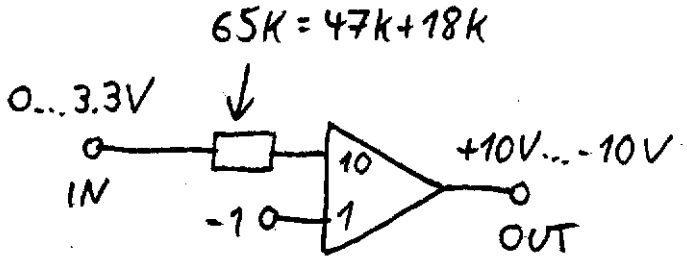
The constants for offset and gain in lines 17, 19, 21 and 23 may need some fine tuning, because the microcontroller uses its own 3.3V supply as analog reference.

For a better software with much more features, see next chapter.

At the top left of the screenshot you see the source code. At the bottom left you see the serial output. At the right is the serial plotter.
Why did I also write two constants -1 and 1 ? That's because the serial plotter does automatically scale the vertical axis to the minimum and maximum in the data, which I found quite disturbing. Writing -1 and 1 effectively disables this autoscale feature.



4.5.3 How to convert 0 to 3.3V Signals to +-1 Machine Unit



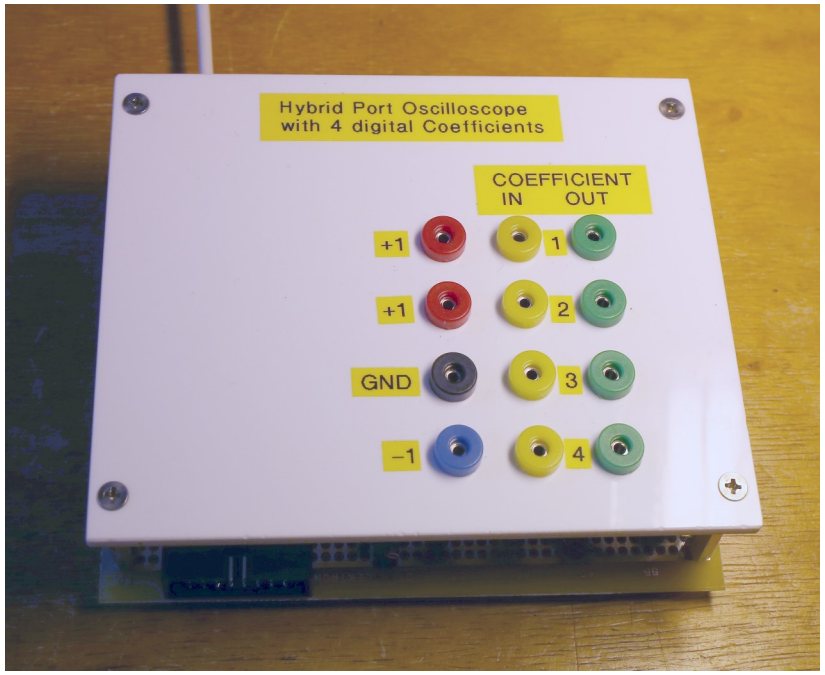
DAC	IN	OUT
0	0V	+10V
512	1,65V	0V
1023	3,3V	-10V

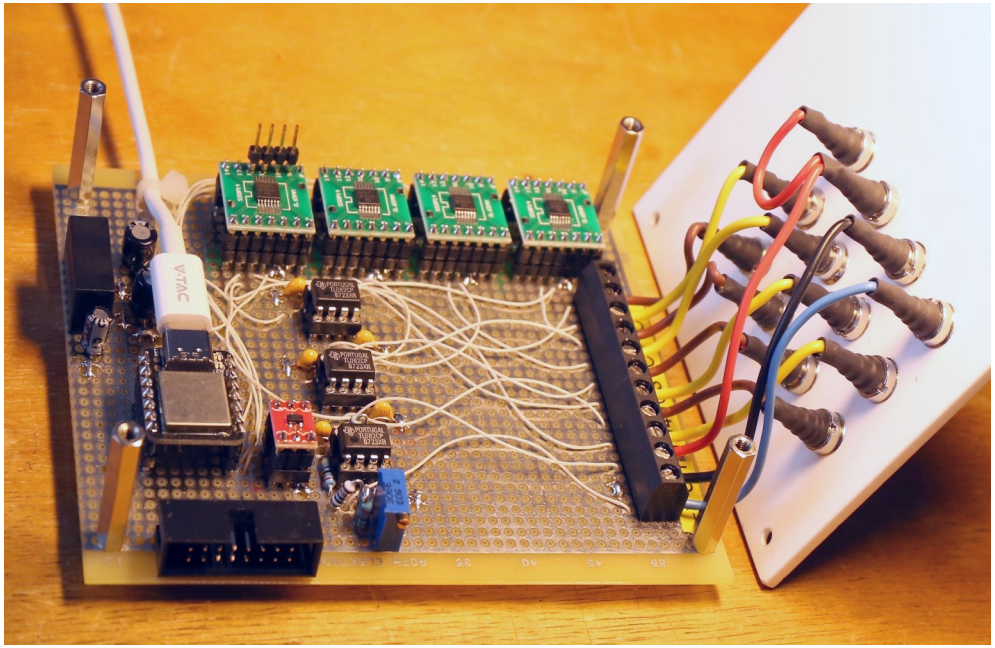
For unipolar 10V to 0V output, use 230k = 220k + 10k as external resistor connected to the 10" input.

For unipolar 0V to -10V output, use 230k = 220k + 10k as external resistor connected to the 10" input and don't use the "1" input.

4.5.4 Hybrid Port Oscilloscope with 4 digital Coefficients

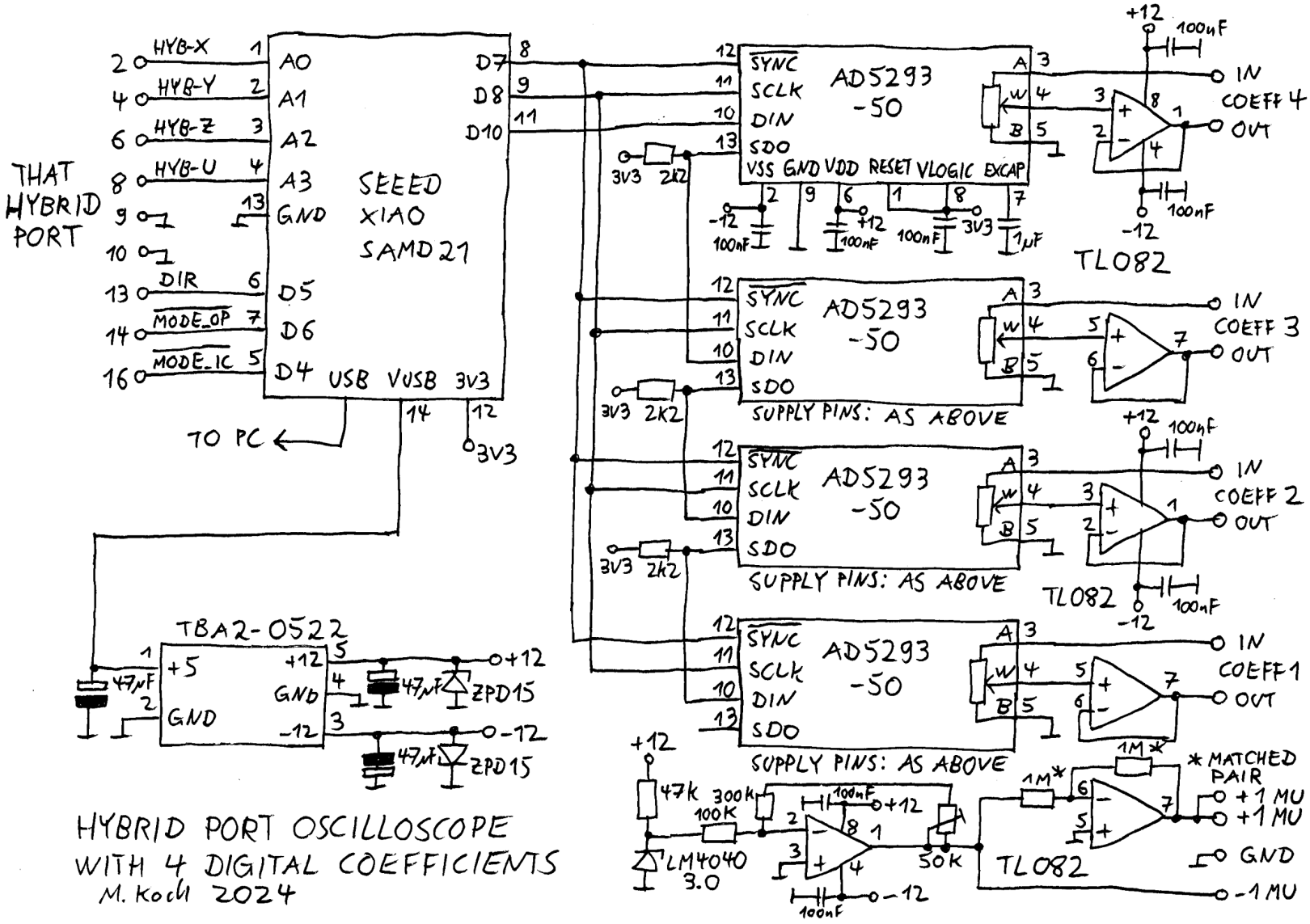
This is the same hybrid port oscilloscope as above, but with four digital coefficients added.





2mm jacks from Aliexpress: https://de.aliexpress.com/item/4000271665708.html

Prototype board with groundplane: https://www.reichelt.de/hf-europlatine-epoxyd-verzinnt-160x100mm-re-201lf-p34772.html



HYBRID PORT OSCILLOSCOPE
WITH 4 DIGITAL COEFFICIENTS
M. Koch 2024

Below is the listing of the software with many features. The same software can also be used for the previous circuit (without digital coefficients):

- **Samplerate from 1Hz to 40kHz (20kHz with 2 channels, 10kHz with 4 channels)**
- **1, 2 or 4 channels**
- **50 to 2000 samples per channel**
- **Oversampling, the ADCs are always running with 40kHz (20kHz with 2 channels, 10kHz with 4 channels)**
- **Full remote control of THAT: IC, OP, HALT**
- **Controlled via serial port from Arduino IDE**
- **Supports the "Serial plotter" in Arduino IDE (which shows only 50 samples per channel, but this number can be increased by a dirty hack)**
- **Built-in help function lists all commands, press "?"**

```
/*
  Oscilloscope for THATs hybrid port with Seeed_XIAO SAMD21
  with 4 digital potentiometers
  Michael Koch, January 2024
*/

#include <TimerTC3.h>
#include <SPI.h>

int channels, samplerate, samples, oversampling;
int ch, ch1, ch2, ch3, ch4, samplesToAcquire;
int ovs1, ovs2, ovs3, ovs4;
int coeff[4];
bool hybrid, serialPlotter, labels;
unsigned short waveform[8000];
int writePointer;

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(D7,OUTPUT);
  digitalWrite(D7,HIGH);   // CS = high
  pinMode(D8,OUTPUT);
  pinMode(D10,OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  analogReadResolution(12);
```

```
  Serial.setTimeout(4294967295);   // about 50 days

  channels = 4;     // initial settings
  samples = 500;
  samplesToAcquire = 0;
  samplerate = 5000;
  hybrid = 1;
  serialPlotter = 1;
  labels = 1;
  oversampling = 40000 / samplerate / channels;

  coeff[0] = coeff[1] = coeff[2] = coeff[3] = 0;

  if (hybrid)
  {
    pinMode(D5,OUTPUT);
    digitalWrite(D5,LOW);   // DIR = low
    pinMode(D4,OUTPUT);
    digitalWrite(D4,HIGH);   // Mode_IC = high
    pinMode(D6,OUTPUT);
    digitalWrite(D6,HIGH);   // Mode_OP = high
  }

  SPI.begin ();
  SPI.setClockDivider(SPI_CLOCK_DIV2);  // divide the clock by 2
  SPI.setDataMode(SPI_MODE1);

  digitalWrite(D7,LOW);   // CS = low   // disable the write protection
  SPI.transfer16(0x1802);               //  in all four AD5293
  SPI.transfer16(0x1802);
  SPI.transfer16(0x1802);
  SPI.transfer16(0x1802);
  digitalWrite(D7,HIGH);  // CS = high
  delayMicroseconds(1);
  digitalWrite(D7,LOW);   // CS = low   // set all four AD5293 to zero
  SPI.transfer16(0x0400);
  SPI.transfer16(0x0400);
  SPI.transfer16(0x0400);
  SPI.transfer16(0x0400);
  digitalWrite(D7,HIGH);  // CS = high

  TimerTc3.initialize(25);   // 25 us, 40kHz
  TimerTc3.attachInterrupt(timerIsr);
```

```
}

void timerIsr()   // 40kHz timer interrupt service routine
{
  if (samplesToAcquire == 0)
    digitalWrite(LED_BUILTIN,HIGH);   // orange LED off
  else
  {
    switch (ch)
    {
      case 0: ch1 += analogRead(A0);
              if(--ovs1 == 0)
              {
                ovs1 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch1 / oversampling);
                ch1 = 0;
              }
              break;
      case 1: ch2 += analogRead(A1);
              if(--ovs2 == 0)
              {
                ovs2 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch2 / oversampling);
                ch2 = 0;
              }
              break;
      case 2: ch3 += analogRead(A2);
              if(--ovs3 == 0)
              {
                ovs3 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch3 / oversampling);
                ch3 = 0;
              }
              break;
      case 3: ch4 += analogRead(A3);
              if(--ovs4 == 0)
              {
                ovs4 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch4 / oversampling);
                ch4 = 0;
              }
              break;
    }
```

```
    samplesToAcquire -= 1;
    ch = (ch+1) % channels;   // 1 channel: ch=0    2 channels: ch=0,1    4 channels: ch=0,1,2,3
  }
}

void acquire()
{
  writePointer = 0;    // set the initial conditions for a new acquisition
  ovs1 = ovs2 = ovs3 = ovs4 = oversampling;
  ch1 = ch2 = ch3 = ch4 = 0;
  ch = 0;

  if (hybrid)
  {
    digitalWrite(D4,LOW);     // Mode_IC = low
    digitalWrite(D6,HIGH);    // Mode_OP = high
    delay(10);                // wait 10ms
    digitalWrite(D4,HIGH);    // Mode_IC = high
    digitalWrite(D6,LOW);     // Mode_OP = low
  }
  digitalWrite(LED_BUILTIN,LOW);   // orange LED on
  samplesToAcquire = samples * oversampling * channels;   // this starts the acquisition

  for( ; samplesToAcquire != 0; )   // wait for end of acquisition
    delay(1);

  int index = 0;
  for (int n=0; n < samples; n++)   // print the waveforms
  {
    for (int j = 0; j < channels; j++)
    {
      if (labels)
      {
        switch (j)
        {
          case 0: Serial.print("X:");
                  break;
          case 1: Serial.print("Y:");
                  break;
          case 2: Serial.print("Z:");
                  break;
          case 3: Serial.print("U:");
                  break;
```

```
        }
      }
      Serial.print(String(-1.656 + 0.000809 * waveform[index++],3) + " "); // offet and gain constants can be fine-tuned
    }
    if(serialPlotter)
      if (labels)
        Serial.print("MIN:-1 MAX:1");
      else
        Serial.print("-1 1");
    Serial.println();
  }
}

// the loop function runs over and over again forever
void loop()
{
  String arg1, arg2, arg3;

  String s = Serial.readStringUntil('\n');
  s.toUpperCase();
  s.trim();
  int i = s.indexOf(' ');
  arg1 = s.substring(0,i);
  if (i!=-1)
    arg2 = s.substring(i);
  else
    arg2 = "";
  arg1.trim();
  arg2.trim();

  if (arg1 == "")      // Acquire
    acquire();

  if (arg1 == "L")     // List settings
  {
    Serial.print(" Channels: ");
    Serial.println(channels);
    Serial.print(" Samples per channel: ");
    Serial.println(samples);
    Serial.print(" Samplerate: ");
    Serial.print(samplerate);
    Serial.println(" Hz");
    Serial.print(" Oversampling factor: ");
```

```
    Serial.println(oversampling);
    Serial.print(" Coefficient 1: ");
    Serial.println((float)coeff[0] / 1024.0, 3);
    Serial.print(" Coefficient 2: ");
    Serial.println((float)coeff[1] / 1024.0, 3);
    Serial.print(" Coefficient 3: ");
    Serial.println((float)coeff[2] / 1024.0, 3);
    Serial.print(" Coefficient 4: ");
    Serial.println((float)coeff[3] / 1024.0, 3);
    Serial.print(" Hybrid mode: ");
    Serial.println(hybrid ? "on" : "off");
    Serial.print(" Arduino serial plotter mode: ");
    Serial.println(serialPlotter ? "on" : "off");
    Serial.print(" Label mode: ");
    Serial.println(labels ? "on" : "off");
    Serial.println();
  }

  if (arg1 == "CH")     // Set number of channels
  {
    switch (arg2.toInt())
    {
      case 1: channels = 1;
              Serial.println("1 Channel");
              break;
      case 2: if (samplerate < 40000)
              {
                channels = 2;
                Serial.println("2 Channels");
              }
              else
              {
                Serial.println("Error: Samplerate is too high");
              }
              break;
      case 4: if (samplerate < 20000)
              {
                channels = 4;
                Serial.println("4 Channels");
              }
              else
              {
                Serial.println("Error: Samplerate is too high");
```

```
            }
            break;
    default: Serial.println("Error: Number of channels must be 1, 2 or 4");
  }
  oversampling = 40000 / samplerate / channels;
}

if (arg1 == "HY")     // Set hybrid mode on / off
{
  switch (arg2.toInt())
  {
    case 0: hybrid = 0;
            pinMode(D4,INPUT);
            pinMode(D5,INPUT);
            pinMode(D6,INPUT);
            Serial.println("Hybrid mode off");
            break;
    case 1: hybrid = 1;
            pinMode(D5,OUTPUT);
            digitalWrite(D5,LOW);   // DIR = low
            pinMode(D4,OUTPUT);
            digitalWrite(D4,HIGH);   // Mode_IC = high
            pinMode(D6,OUTPUT);
            digitalWrite(D6,HIGH);   // Mode_OP = high
            Serial.println("Hybrid mode on");
            break;
    default: Serial.println("Error: Argument must be 0 or 1");
  }
}

if (arg1 == "SP")     // Set serial plotter mode on / off
{
  switch (arg2.toInt())
  {
    case 0: serialPlotter = 0;
            Serial.println("Serial plotter mode off");
            break;
    case 1: serialPlotter = 1;
            Serial.println("Serial plotter mode on");
            break;
    default: Serial.println("Error: Argument must be 0 or 1");
  }
}
```

```
if (arg1 == "LA")     // Set label mode on / off
{
  switch (arg2.toInt())
  {
    case 0: labels = 0;
            Serial.println("Label mode off");
            break;
    case 1: labels = 1;
            Serial.println("Label mode on");
            break;
    default: Serial.println("Error: Argument must be 0 or 1");
  }
}

if (arg1 == "IC")     // Set THAT to IC mode
{
  if (hybrid)
  {
    digitalWrite(D4,LOW);     // Mode_IC = low
    digitalWrite(D6,HIGH);    // Mode_OP = high
    delay(10);
    Serial.println("THAT set to IC mode");
  }
  else
    Serial.println("Error: THAT is not in hybrid mode");
}

if (arg1 == "OP")     // Set THAT to OP mode
{
  if (hybrid)
  {
    digitalWrite(D4,HIGH);    // Mode_IC = high
    digitalWrite(D6,LOW);     // Mode_OP = low
    Serial.println("THAT set to OP mode");
    acquire();
  }
  else
    Serial.println("Error: THAT is not in hybrid mode");
}

if (arg1 == "HA")     // Set THAT to HALT mode
{
```

```
  if (hybrid)
  {
    digitalWrite(D4,HIGH);    // Mode_IC = high
    digitalWrite(D6,HIGH);    // Mode_OP = high
    Serial.println("THAT set to HALT mode");
  }
  else
    Serial.println("Error: THAT is not in hybrid mode");
}

if (arg1 == "SA")     // Set number of samples
{
  int c = arg2.toInt();
  if ((c >= 50) && (c <= 2000))
  {
    samples = c;
    Serial.print(samples);
    Serial.println(" Samples");
  }
  else
    Serial.println("Error: Number of samples must be between 50 and 2000");
}

if (arg1 == "CO")     // Set coefficient
{
  int i = arg2.indexOf(' ');
  arg3 = arg2.substring(i);
  arg2 = arg2.substring(0,i);
  arg2.trim();
  arg3.trim();
  int co_nr = arg2.toInt() - 1;
  if ((co_nr < 0) || (co_nr > 3))
    Serial.println("Error: Coefficient number must be 1, 2, 3 or 4");
  else
  {
    int c = (int)(1024 * arg3.toFloat());
    if (c < 0) c = 0;
    if (c > 1023) c = 1023;
    coeff[co_nr] = c;
    digitalWrite(D7,LOW);   // CS = low
    SPI.transfer16(0x0400 + coeff[0]);   // write data to all 4 AD5293
    SPI.transfer16(0x0400 + coeff[1]);
    SPI.transfer16(0x0400 + coeff[2]);
```

```
    SPI.transfer16(0x0400 + coeff[3]);
    digitalWrite(D7,HIGH);   // CS = high
    Serial.print(" Coefficient ");
    Serial.print(co_nr + 1);
    Serial.print(" was set to ");
    Serial.println((float)c / 1024.0);
  }
}

if (arg1 == "SR")     // Set samplerate
{
  int c = arg2.toInt();
  switch (c)
  {
    case 1:
    case 2:
    case 5:
    case 10:
    case 20:
    case 50:
    case 100:
    case 200:
    case 500:
    case 1000:
    case 2000:
    case 5000:
    case 10000: samplerate = c;
                break;
    case 20000: if (channels != 4)
                  samplerate = c;
                else
                  Serial.println("Error: Samplerate 20000 isn't supported with 4 channels");
                break;
    case 40000: if (channels == 1)
                  samplerate = c;
                else
                  Serial.println("Error: Samplerate 40000 isn't supported with 2 or 4 channels");
                break;
    default: Serial.println("Error: Invalid samplerate");
  }
  oversampling = 40000 / samplerate / channels;
  Serial.print(" Samplerate: ");
  Serial.print(samplerate);
```

```
  Serial.println(" Hz");
}

if (arg1 == "?")
{
  Serial.println("*********************************************************************************");
  Serial.println("THAT Hybrid Port Oscilloscope with Seeed XIAO SAMD21   (C) Michael Koch 2024");
  Serial.println("*********************************************************************************");
  Serial.println("");
  Serial.println("List of possible samplerates and oversampling factors:");
  Serial.println("Samplerate  Sample  Oversampling Factor");
  Serial.println("     Hz    Interval   1Ch   2Ch   4Ch ");
  Serial.println("   40000     25us      1     -     - ");
  Serial.println("   20000     50us      2     1     - ");
  Serial.println("   10000    100us      4     2     1 ");
  Serial.println("    5000    200us      8     4     2 ");
  Serial.println("    2000    500us     20    10     5 ");
  Serial.println("    1000      1ms     40    20    10 ");
  Serial.println("     500      2ms     80    40    20 ");
  Serial.println("     200      5ms    200   100    50 ");
  Serial.println("     100     10ms    400   200   100 ");
  Serial.println("      50     20ms    800   400   200 ");
  Serial.println("      20     50ms   2000  1000   500 ");
  Serial.println("      10    100ms   4000  2000  1000 ");
  Serial.println("       5    200ms   8000  4000  2000 ");
  Serial.println("       2    500ms  20000 10000  5000 ");
  Serial.println("       1      1s   40000 20000 10000 ");
  Serial.println("");
  Serial.println("List of commands:");
  Serial.println("?       This help page");
  Serial.println("IC      Set THAT to IC mode and wait 10ms");
  Serial.println("         This works only if THAT is in hybrid mode, see HY command");
  Serial.println("OP      Set THAT to OP mode, and start data acquisition");
  Serial.println("         This works only if THAT is in hybrid mode, see HY command");
  Serial.println("HA      Set THAT to HALT mode");
  Serial.println("         This works only if THAT is in hybrid mode, see HY command");
  Serial.println("<cr>    If THAT is in hybrid mode: Set THAT to IC, wait 10ms, then set THAT to OP and start data acquisition");
  Serial.println("        If THAT is not in hybrid mode: Start data acquisition");
  Serial.println("CH n    Set the number of channels, n = 1, 2 or 4, default is 4");
  Serial.println("         2 channels are only possible with samplerate 20000 or slower");
  Serial.println("         4 channels are only possible with samplerate 10000 or slower");
  Serial.println("SA n    Set the number of samples per channel, n = 50 to 2000, default is 500");
  Serial.println("         By default the Arduino IDE's serial plotter shows only 50 samples per channel");
```

```
    Serial.println("         But it's possible to change it to 2000 with a dirty hack");
    Serial.println("SR n    Set samplerate in Hz, n = 1 to 40000 (see table above), default is 1000");
    Serial.println("         Samplerate 20000 is only possible with 1 or 2 channels");
    Serial.println("         Samplerate 40000 is only possible with 1 channel");
    Serial.println("         The internal samplerate is always as high as possible for oversampling");
    Serial.println("CO n x  Set coefficient n (1,2,3,4) to the value x (0.000 to 1.000)");
    Serial.println("           The resolutiuon is 10-bit or about 0.001");
    Serial.println("HY b    Set THAT to hybrid mode, can be 0 (off) or 1 (on), default is 1");
    Serial.println("         0 means THAT is controlled by its MODE switch");
    Serial.println("         1 means THAT is controlled via hybrid port");
    Serial.println("SP b    Select serial plotter mode, can be 0 (off) or 1 (on), default is 1");
    Serial.println("         1 means two dummy channels with constants -1 and 1 are added,");
    Serial.println("         so that the serial plotter's autoscale feature is disabled");
    Serial.println("LA b    Label mode, can be 0 (off) or 1 (on), default is 1");
    Serial.println("         1 means the signals are labeled X, Y, Z, U, MIN and MAX");
    Serial.println("         0 means no labels are sent so the serial plotter");
    Serial.println("L       List the actual settings");
    Serial.println("");
  }
}
```

4.5.5 Software for serial plotting

The "Serial plotter" in Arduino IDE V2.2.1 displays only 50 points on the x axis by default.

The older version V1.8.18 did show 500 points, but had a [-5 ... +5] or [-6 ... +6] range for the y axis.

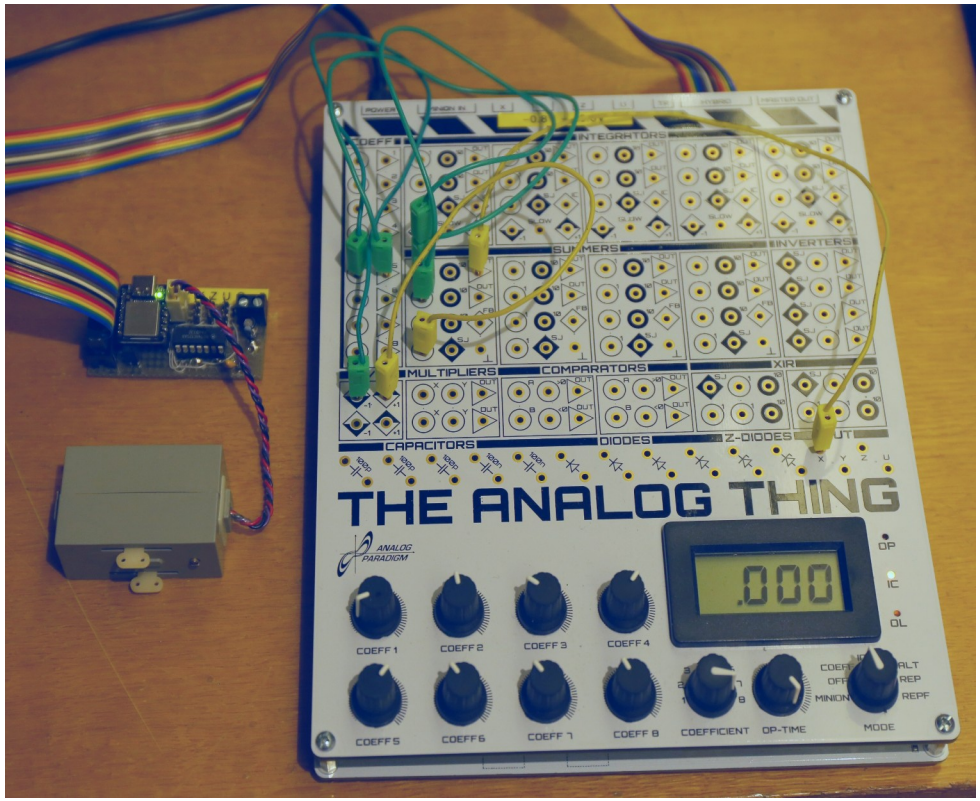
Workaround for showing more than 50 samples in the serial plotter in Arduino IDE V2.2.1:
This file must be hacked:
C:\Program Files\Arduino IDE\resources\app\lib\backend\resources\arduino-serial-plotter-webapp\static\js\main.35ae02cb.chunk.js
Before you can edit this file, you must change the security properties of the file, and allow write access by the user.
Then open the file in Notepad, search for
U=Object(o.useState)(50)
and replace it by
U=Object(o.useState)(2000) (or use any other number of samples)
Save the file.
Now restart the Arduino IDE and the serial plotter will show 2000 samples!
Most probably this hack must be repeated when the Arduino IDE is updated to a new version.
I found this hack here (thank you Alex Tsyganok!), but on my computer the path was different than he described:
https://github.com/arduino/arduino-ide/issues/803#issuecomment-1338149431

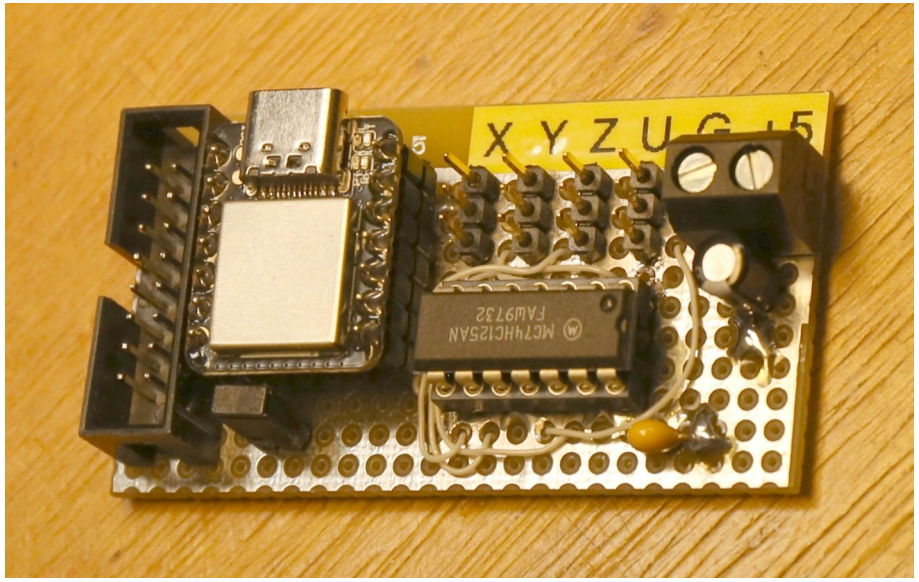
Better Serial Plotter (freeware), unfortunately it does only receive data and it's not possible to send commands to the microcontroller:
https://hackaday.io/project/181686-better-serial-plotter

This "Processing Grapher" project might also be useful: https://wired.chillibasket.com/processing-grapher/

4.5.6 Controlling Servos with the Hybrid Port

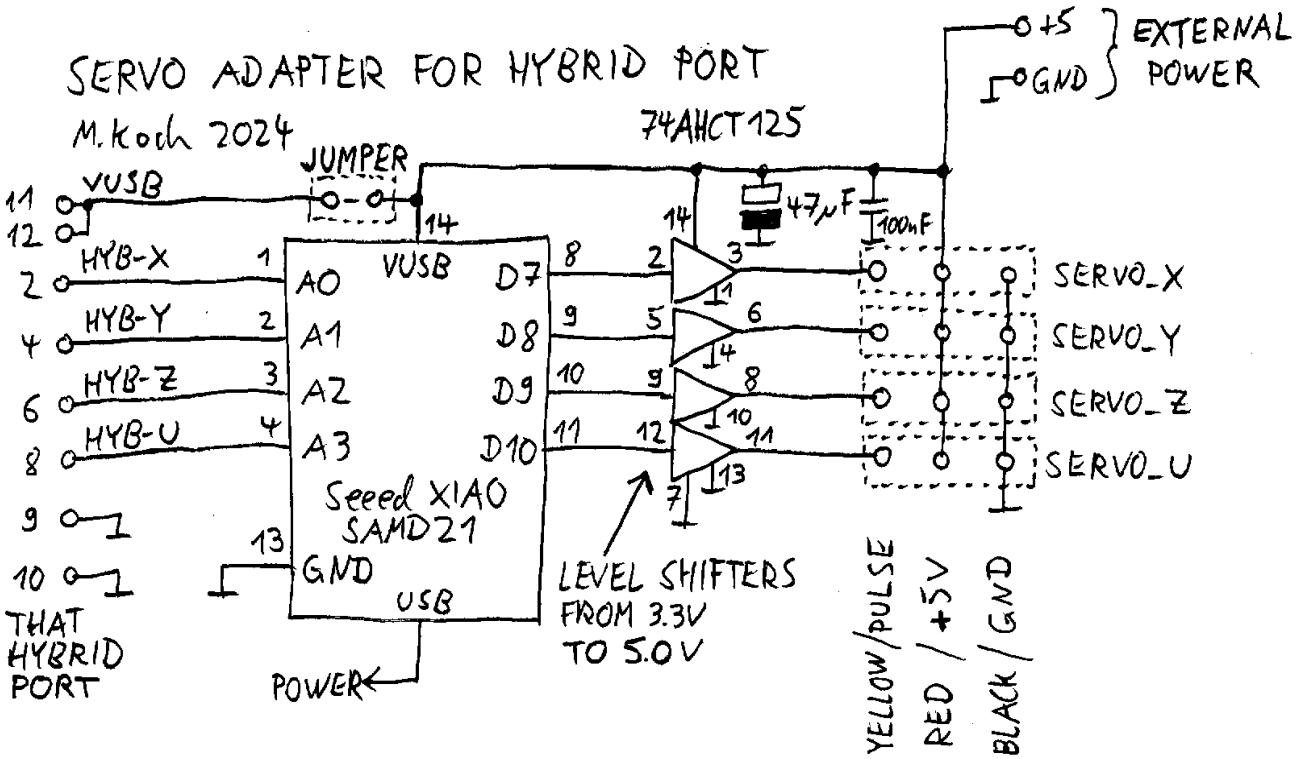
This is an adapter for controlling four servos from the hybrid port. The four servo's positions are proportional to the X, Y, Z and U signals.





Power for the servos can come from three sources:

- **From THATs hybrid port, in this case install the jumper and don't connect a USB cable to the SAMD21 controller, and don't connect external power. This is only suitable for small servos, because THAT has an internal 750mA PTC fuse.**
- **From SAMD21 USB port, in this case remove the jumper and don't connect external power. This is only suitable for small servos, because the USB port can only supply 1000mA.**
- **From external power connector, in this case remove the jumper and don't connect a USB cable to the SAMD21 controller. This is also suitable for large servos. The external power connector can also be used for servos which need only 4.2V supply voltage.**



See also:https://www.arduino.cc/reference/en/libraries/servo/

Cheap servos: https://de.aliexpress.com/item/1005005857192248.html

Software:

```
/*
  THAT Hybrid Port Adapter for 4 Servos
  Michael Koch, January 2024
*/

#include <Servo.h>

Servo myservo1, myservo2, myservo3, myservo4;
int minimum, maximum, mean, ampl;
float gain, offset;

// the setup function runs once when you press reset or power the board
void setup()
{
  analogReadResolution(12);
  minimum = 500;            // minimum pulse width in us
  maximum = 2500;           // maximum pulse width in us
  mean = (minimum + maximum) / 2;
  ampl = maximum - mean;
  gain = 0.000809;          // gain
  offset = -1.656;          // offset
  myservo1.attach(D7,minimum,maximum);
  myservo2.attach(D8,minimum,maximum);
  myservo3.attach(D9,minimum,maximum);
  myservo4.attach(D10,minimum,maximum);
}

// the loop function runs over and over again forever
void loop()
{
  float f;
  f = offset + gain * analogRead(A0);                  // -1 to +1
  myservo1.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  f = offset + gain * analogRead(A1);                  // -1 to +1
  myservo2.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  f = offset + gain * analogRead(A2);                  // -1 to +1
  myservo3.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  f = offset + gain * analogRead(A3);                  // -1 to +1
  myservo4.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  delay(10);
}
```

List of servos (sorted by speed):

Servo Type	Voltage [V]	Current [A]	Torque [kg·cm]	Speed [s/60°]	Weight [g]	Notes and Links
Align DS825	7.4-8,4		10 @7.4V 12.5 @8.4V	0.030 @7.4V 0.020 @8.4V	62 or 72?	https://www.flashrc.com/de/servos/24744-align-ds825-brushless-hv-digital-servo-hsd82502-62g-12-5kg-cm-0-02s-60.html https://www.freakware.de/p/ds825m-high-voltage-brushless-servo-align-hsd82501-a147730.htm
ST4015	-8.4		15.5	0.020 @8.4V	75	https://www.modell-hubschrauber.at/Elektronisches-Zubehoer/Servos/Servos/Heckservos/Standard-Heckservos/1st-Brushless-Heckservo-ST-4015MG-HV-Digital::42675.html
Savöx SB-2272MG+	4.8-8.4		7 @6.0V 10 @8.4V	0.032 @6V 0.028 @8.4V	69	https://www.savox-shop.com/de/savoex-sb-2272mg-servo.html
DS835M	7.0-8.4		10.0 @7.4V 12.5 @8.4V	0.030 @7.4V 0.028 @8.4V	72	https://www.freakware.de/p/ds835m-high-voltage-brushless-servo-align-hsd83501-a177453.htm
BL855H	6.0-8.4		8 @6.0V 10 @7.4V 12.5 @8.4V	0.040 @6.0V 0.030 @7.4V 0.028 @8.4V	73	https://www.freakware.de/p/bl855h-high-voltage-brushless-servo-align-hsl85501-a106191.htm
Yantrs 0318AS-MGX-E	6.0-8.4	6.0	12 @4.2A 16 @5.5A 18 @6.0A	0.040 @6.0V 0.035 @7.4V 0.030 @8.4V	70	https://yantrs.cn/products/yantrs-helical-0-04s-27kg-low-profile-rc-servo-all-metal-waterproof-brushless-servo-%E7%9A%84%E5%89%AF%E6%9C%AC
DS835	6.0-8.4		5.0 @6.0V 6.5 @8.4V	0.050 @6.0V 0.035 @8.4V	29	https://www.freakware.de/p/ds535-digital-servo-align-hsd53502-a146314.htm
BH9012	6.0-8.4	3.3	8.5 @6.0V 9.5 @7.4V 12.0 @8.4V	0.050 @6.0V 0.040 @7.4V 0.035 @8.4V	70	https://moduapp.de/shop/shop/srt-brushless-servo-hv-high-speed-9-5kg-0-04sec-7-4v/
HiTEC HSG-8315BH	6.0-7.4		4.3 @6.0V 5.3 @7.4V	0.050 @6.0V 0.040 @7.4V	60	https://www.freakware.de/p/hitec-hs-8315bh-hit-114315-a85753.htm
Yantrs 0427AS-MGX-E	6.0-8.4	6.0	20 @6.0V 23 @7.4V 27 @8.4V	0.058 @6.0V 0.047 @7.4V 0.040 @8.4V	70	https://de.aliexpress.com/item/1005005224149218.html

Servo Type	Voltage [V]	Current [A]	Torque [kg·cm]	Speed [s/60°]	Weight [g]	Notes and Links
GX_Servo X15BLS	6.0-8.4		12 @6.0V 15 @7.4V 18 @8.4V	0.070 @6.0V 0.060 @7.4V 0.050 @8.4V	70	https://de.aliexpress.com/item/1005004649121319.html
GX_Servo X20BLS			18 @6.0V 20 @7.4V 22 @8.4V	0.060 @6.0V 0.050 @7.4V 0.045 @8.4V		
GX_Servo X25BLS			22 @6.0V 25 @7.4V 28 @8.4V	0.060 @6.0V 0.050 @7.4V 0.045 @8.4V		
Savöx SH-1290MG	4.8-6.0		4.0 @4.8V 5.0 @6.0V	0.059 @4.8V 0.048 @6.0V	57	https://www.savox-shop.com/de/savoex-sh-1290mg-servo.html
Real Hawk?	3.7-4.2	0.035	0.065 @3.7V 0.075 @4.2V	0.070 @3.7V 0.050 @4.2V	1.9	ultra small https://de.aliexpress.com/item/33006571747.html
A06CLS	4.8-8.4		1.8 @4.8V 2.2 @6.0V 2.6 @7.4V 3.0 @8.4V	0.075 @4.8V 0.066 @6.0V 0.058 @7.4V 0.052 @8.4V	7	ultra small https://de.aliexpress.com/item/1005005104844054.html
DS820M	7.4-8.4		22 @7.4V 23 @8.4V	0.060 @7.4V 0.055 @8.4V	80	https://www.freakware.de/p/ds820m-high-voltage-brushless-servo-align-hsd82001-a147728.htm
Yantrs 0640AS-MGX-E	6.0-8.4	6.0	32 @6.0V 37 @7.4V 40 @8.4V	0.080 @6.0V 0.070 @7.4V 0.060 @8.4V	70	https://de.aliexpress.com/item/1005005423898270.html
Yantrs 0612MG-B	6.0-8.4	2.8	9 @6.0V 10.5 @7.4V 12 @8.4V	0.090 @6.0V 0.070 @7.4V 0.060 @8.4V	22	https://de.aliexpress.com/item/1005005381188231.html
Sirenxi?	3.7-6.0		0.2 @4.8V	0.080 @4.8V	3.1	ultra small https://de.aliexpress.com/item/1005005652463467.html
DS3240	4.8-6.8		36 @5.0V 45 @6.8V	0.200 @5.0V 0.170 @6.8V	68.3	https://de.aliexpress.com/item/1005005857192248.html

Servo Type	Voltage [V]	Current [A]	Torque [kg·cm]	Speed [s/60°]	Weight [g]	Notes and Links
SG90	4.2-6.0		1.6 @4.8V	0.300 or 0.120 ?	10.1	very cheap https://de.aliexpress.com/item/1005005857192248.html
ASMC-03B	11-30	?	180	0.50 @24V	530	ultra high torque https://de.aliexpress.com/item/4000322402143.html
ASMC-04A	11-30	3.0	110	0.12 @24V	560	ultra high torque https://de.aliexpress.com/item/1005006010596284.html
ASMC-04B			180	0.50 @24V		
ASMC-05A	11-30	3.0	110 @24V	0.12 @24V	530	ultra high torque https://de.aliexpress.com/item/32889090668.html https://de.aliexpress.com/item/32820735038.html
ASMC-05B			180 @24V	0.50 @24V		
ASMC-LQB	8-12	1.0 ?	60	0.50 ?	170	https://de.aliexpress.com/item/1005004137170731.html https://de.aliexpress.com/item/1005005412800200.html
ASME-02						https://content.instructables.com/FCA/QIQI/IQA4XCJY/FCAQIQIIQA4XCJY.pdf
ASME-04A	11-30	5.0	220 @12V	0.12 @24V	560	ultra high torque https://de.aliexpress.com/item/1005006010596284.html
ASME-04B			340 @24V	0.50 @24V		
ASME-05B	11-30	5.0	380 @24V	0.50 @24V	560	ultra high torque https://de.aliexpress.com/item/32973185578.html
ASME-MRA	11-30	5.0	110 or 260?	0.12 @24V	560	ultra high torque https://de.aliexpress.com/item/1005005633621592.html https://de.aliexpress.com/item/1005005417403122.html https://de.aliexpress.com/item/1005004126382458.html
ASME-MRB			380 or 400?	0.50 @24V		
ASME-MXA	11-30	5.0	130 @12V 260 @24V	0.24 @12V 0.12 @24V	550	ultra high torque with 10 revolutions https://de.aliexpress.com/item/1005006010596284.html https://de.aliexpress.com/item/1005006236191122.html
ASME-MXB			190 @12V 380 @24V	1.00 @12V 0.50 @24V		
ASME-SDA	7-30	5.0	90	0.12 @24V	450	ultra high torque https://de.aliexpress.com/item/1005004756047692.html
ASME-SDB			180	0.50 @24V		
ASME-SQA	7-24	1.0 ?	90 @24V	0.12 @24V	400	ultra high torque

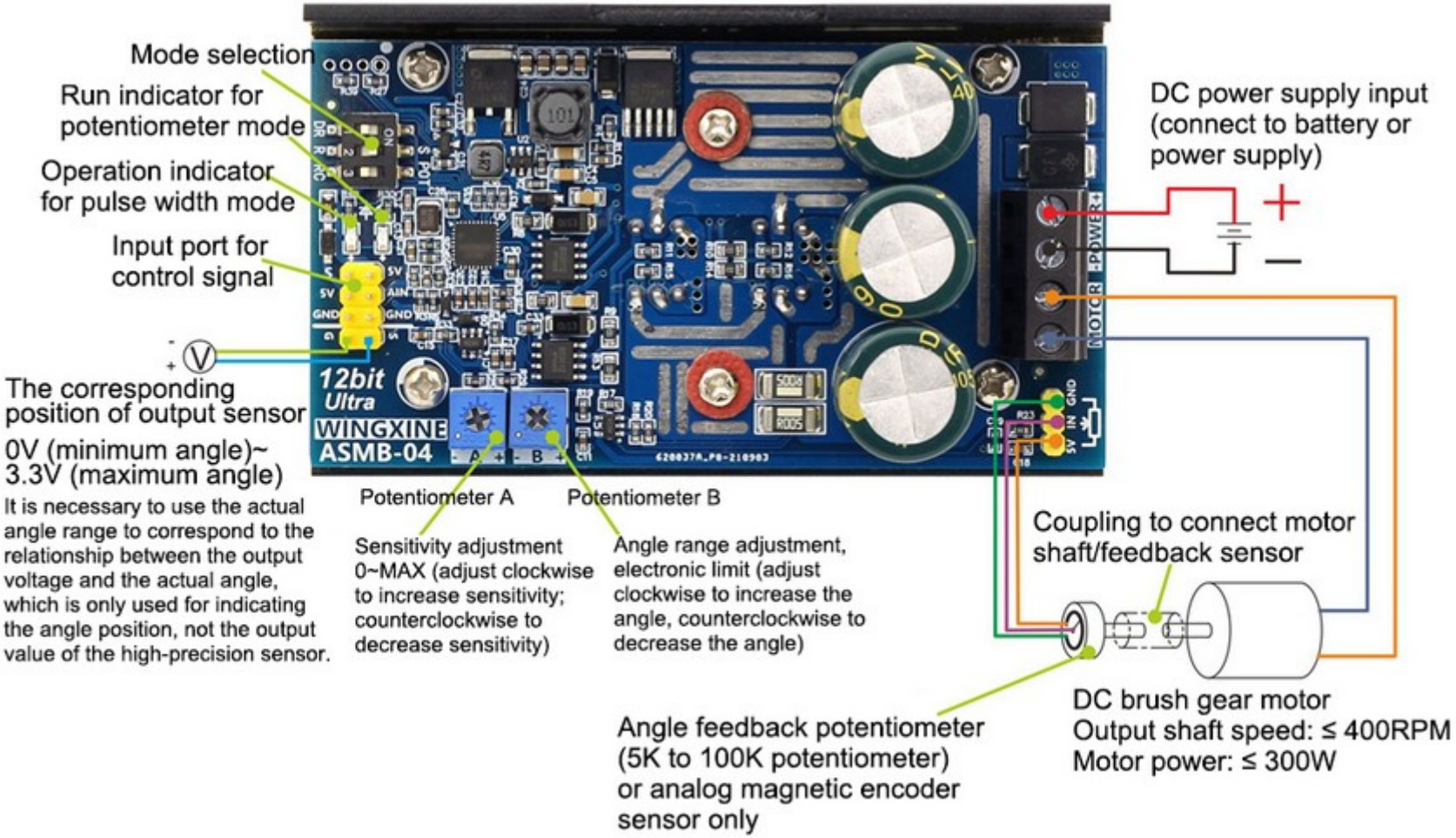
Servo Type	Voltage [V]	Current [A]	Torque [kg·cm]	Speed [s/60°]	Weight [g]	Notes and Links
ASME-SQB			180 @24V	0.50 @24V		https://de.aliexpress.com/item/1005004137302220.html
ASMG-MTA	11-30	10.0	260 @24V	0.12 @24V	550	ultra high torque https://de.aliexpress.com/item/1005005440128075.html https://de.aliexpress.com/item/1005006310483520.html
ASMG-MTB			500 @24V	0.50 @24V		
DHEB-GF775	12-30	5.0	500 @24V	0.50 @24V	1065	ultra high torque, 270° https://de.aliexpress.com/item/1005005552608944.html https://de.aliexpress.com/item/1005005721446943.html
DHLG-03X	24-30	20	700 or 1000 ?	0.50 @24V	4500 or 6300 ?	ultra high torque https://de.aliexpress.com/item/1005004305694514.html https://de.aliexpress.com/item/1005005212645904.html

Hacking the firmware for Wingxine (ASME) servos: https://hackaday.io/project/19082-wingxine-asme-03-alternative-firmware

Servo Controllers:

Type	Voltage [V]	Current [A]	Notes and Links
ASMF-04	7-30	10	with feedback potentiometer https://de.aliexpress.com/item/1005004137043781.html
ASMB-03	8-48	20	with feedback potentiometer https://de.aliexpress.com/item/1005004133955023.html https://www.amazon.de/Okuyonic-Servomotor-Controller-Servomotor-Treiberplatine-Servocontroller-Platine-Servomontage/dp/B09YP7TF94 https://www.alibaba.com/product-detail/DIY-8V-48V-20A-1000N-m_60492358055.html
ASMB-03M	8-48	20	with magnetic feedback sensor https://de.aliexpress.com/item/1005004133955023.html
ASMB-04	8-48	20	with feedback potentiometer https://de.aliexpress.com/item/1005006519077313.html https://www.ebay.de/itm/155281102015

Manual for ASMB-04:



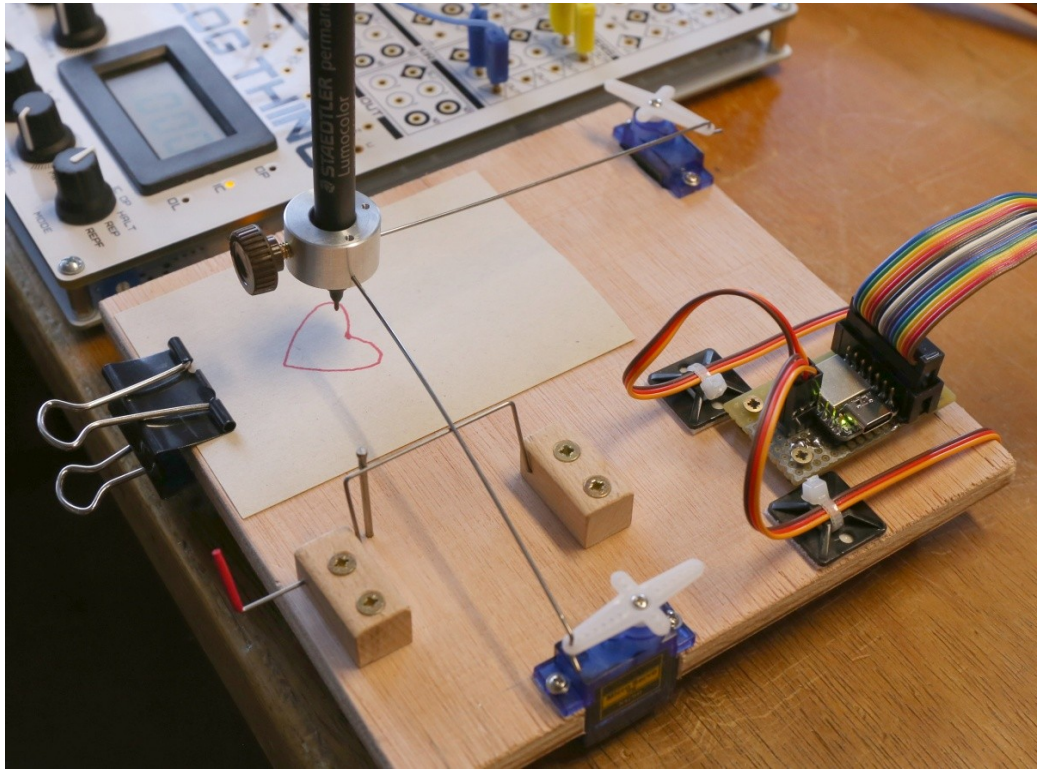
What maximum (sine) frequency can be expected at the servo axis?

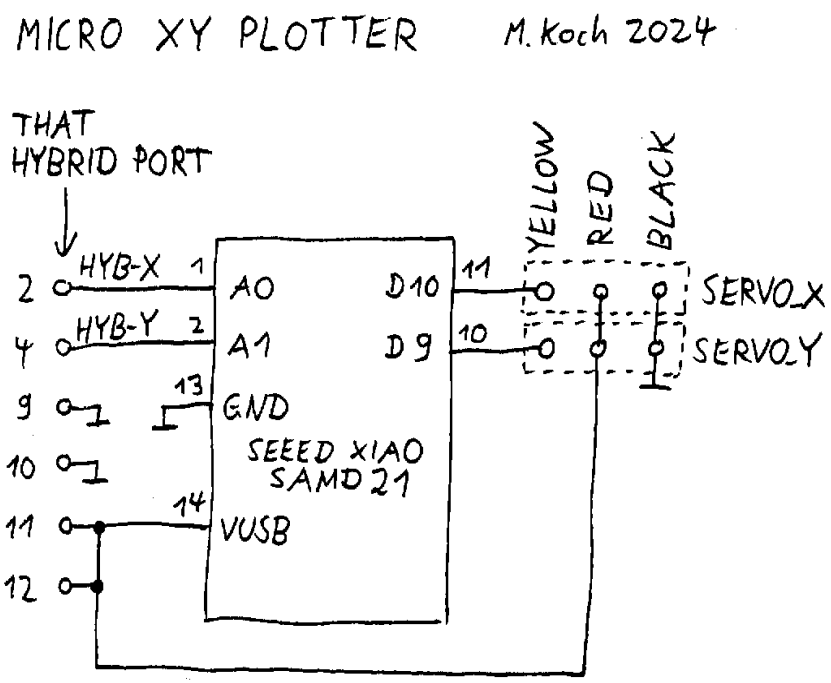
$F_{max} = 1 / (2 \cdot pi \cdot T)$

Time for 60° Movement	F_{max} for 60° Peak Amplitude (120° Peak to Valley)	F_{max} for 30° Peak Amplitude (60° Peak to Valley)
0.02 s	7.96 Hz	15.92 Hz
0.025 s	6.37 Hz	12.73 Hz
0.03 s	5.31 Hz	10.61 Hz
0.035 s	4.55 Hz	9.09 Hz
0.04 s	3.98 Hz	7.96 Hz
0.05 s	3.18 Hz	6.37 Hz
0.06 s	2.65 Hz	5.31 Hz
0.08 s	1.99 Hz	3.98 Hz
0.10 s	1.59 Hz	3.18 Hz
0.20 s	0.80 Hz	1.59 Hz
0.30 s	0.53 Hz	1.06 Hz

4.5.7 Micro XY Plotter for Hybrid Port

This is a very simple XY plotter for the hybrid port, realized with a Seeed XIAO SAMD21 microcontroller and two SG90 servos. It seems that level conversion from 3.3V to 5.0V is not required for these servos. Of couse, the X and Y position of the pen is a nonlinear function of the servo positions. The linearization is done in software, so that the pen position is proportional to the X and Y signals in THAT..





Software for Seeed XIAO SAMD21 microcontroller in XY plotter:

```
//  THAT XY Plotter
//  Michael Koch, January 2024

#include <Servo.h>

Servo myservo1, myservo2;
float x, y, b, c, d, e, xx, yy, alpha, beta;
const float a = 17.2;          // length of servo arm in mm, from servo axis to hinge
const float r = 114;           // length of rod in mm, from pen to hinge
const float p = 13;            // half of drawing range in mm (scale factor mm/MU)
const float gain = 0.000809;   // gain for ADU to MU conversion
const float offset = -1.656;   // offset for ADU to MU conversion

void setup()  // the setup function runs once when you press reset or power the board
{
  analogReadResolution(12);
  myservo1.attach(D10);
  myservo2.attach(D9);
}

void loop()   // the loop function runs over and over again forever
{
  x = offset + gain * analogRead(A0);   // -1 to +1
  y = offset + gain * analogRead(A1);   // -1 to +1
  x = x * p;                            // convert to mm
  y = y * p;                            // convert to mm
  b = (y - a) / (r - x);
  c = (x * x + y * y - 2 * r * r) / (2 * (x - r));
  d = (2 * b * (c - x) - 2 * y) / (1 + b * b);
  e = (x * x - 2 * x * c + c * c + y * y - r * r) / (1 + b * b);
  yy = -d / 2 - sqrt(d * d / 4 - e);    // coordinates of hinge
  xx = yy * b + c;                      //  of X axis servo
  alpha = atan((xx - r) / (a - yy));    // X servo angle
  b = (x + a) / (y - r);
  c = (x * x + y * y - 2 * r * r) / (2 * (y - r));
  d = (2 * b * (c - y) + 2 * x) / (1 + b * b);
  e = (y * y - 2 * y * c + c * c + x * x - r * r) / (1 + b * b);
  yy = -d / 2 - sqrt(d * d / 4 - e);    // coordinates of hinge
  xx = yy * b + c;                      //  of Y axis servo
  beta = atan((xx - r) / (a - yy));     // Y servo angle
  myservo1.writeMicroseconds(1500 + (int)(alpha / 3.1415 * 2000));  // convert from radians to pulse width
  myservo2.writeMicroseconds(1500 + (int)(beta / 3.1415 * 2000));   // convert from radians to pulse width
}
```

How does the linearization work?

The origin of the coordinate system is in the center of the plotting area.

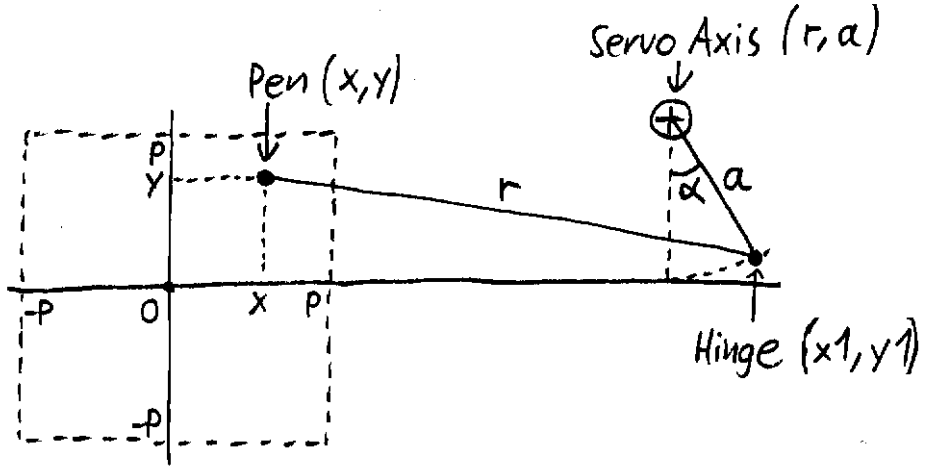
(x,y) is the known pen position.

(r,a) is the known position of the servo axis for the x-axis movement.

a is the length of the servo arm (from axis to hinge).

r is the length from the hinge to the pen.

(x_1,y_1) is the unknown position of the hinge.



We have two given equations:

$(x - x_1)^2 + (y - y_1)^2 = r^2$ **and** $(r - x_1)^2 + (a - y_1)^2 = a^2$

After subtracting the second equation from the first one and solving for x_1, we get:

$x_1 = y_1 \cdot b + c$ **with** $b = (y - a) / (r - x)$ **and** $c = (x^2 + y^2 - 2 \cdot r^2) / (2 \cdot (x - r))$

Insert this x_1 into the first equation to get the solution for y_1:

$y_1 = -d / 2 - \text{sqrt}(d \cdot d / 4 - e)$ **with** $d = (2 \cdot (b \cdot c - x \cdot b - y)) / (1 + b^2)$ **and** $e = (x^2 - 2 \cdot x \cdot c + c^2 + y^2 - r^2) / (1 + b^2)$

and then we get

$x_1 = y_1 \cdot b + c$

The angle of the servo axis is $\text{alpha} = \text{atan}((x_1 - r) / (a - y1))$

For the y-axis servo you can use the same equations after replacing x by y and y by -x.

Pulse width for SG90 servo: 500µs -90°, 1000µs -45°, 1500µs 0°, 2000µs +45°, 2500µs +90°

Calculate the servo angle in three-dimensional space:

The servo arm is in the x,y plane, with the servo axis on the z axis.

a is the length of the servo arm (from axis to hinge).

(x_1,y_1) is the unknown position of the hinge.

(x,y,z) is the known position of the end of the rod.

r is the length of the rod, from the hinge $(x_1,y_1,0)$ to point (x,y,z).

We have two given equations:

$$(x - x_1)^2 + (y - y_1)^2 + z^2 = r^2 \quad \text{and} \quad x_1^2 + y_1^2 = a^2$$

After subtracting the second equation from the first one and solving for x1, we get:

$$x_1 = y_1 \cdot b + c \quad \text{with } b = -y / x \text{ and } c = (x^2 + y^2 + z^2 + a^2 - r^2) / (2 \cdot x)$$

Insert this x1 into the second equation to get the solution for y1:

$$y_1 = +\text{-} \operatorname{sqrt}(a^2 - x_1^2)$$

The angle of the servo axis is:

$$\text{alpha} = \operatorname{atan}(x_1 / y_1) \quad \text{or} \quad \text{alpha} = \operatorname{asin}(x_1 / a)$$

4.5.8 Seeed XIAO Expansion Board, OLED Display and SD Card

How to install the OLED library: https://wiki.seeedstudio.com/How_to_install_Arduino_Library/

Library for OLED display: https://github.com/olikraus/U8g2_Arduino

u8g2 Reference Manual: https://github.com/olikraus/u8g2/wiki/u8g2reference

u8x8 Reference Manual: https://github.com/olikraus/u8g2/wiki/u8x8reference

Wiki: https://github.com/olikraus/u8g2/wiki

Schematic diagram: https://files.seeedstudio.com/wiki/Seeeduino-XIAO-Expansion-Board/document/Seeeduino%20XIAO%20Expansion%20board_v1.0_SCH_200824.pdf

Left button: GND to RESET Right button: GND to D1

U8g2 also includes U8x8 library. Features for U8g2 and U8x8 are:

- **U8g2**
  - **Includes all graphics procedures (line/box/circle draw).**
  - **Supports many fonts. (Almost) no restriction on the font height.**
  - **Requires some memory in the microcontroller to render the display.**
- **U8x8**
  - **Text output only (character) device.**
  - **Only fonts allowed with fixed size per character (8x8 pixel).**
  - **Writes directly to the display. No buffer in the microcontroller required.**

SD Library: https://www.arduino.cc/reference/en/libraries/sd/

The SD library supports FAT16 and FAT32 file systems on standard SD cards and SDHC cards. It uses short 8.3 names for files. The file names passed to the SD library functions can include paths separated by forward-slashes, /, e.g. “directory/filename.txt”. Because the working directory is always the root of the SD card, a name refers to the same file whether or not it includes a leading slash (e.g. “/file.txt” is equivalent to “file.txt”).

Example for listing the directory of a SD card: https://docs.arduino.cc/learn/programming/sd-guide/

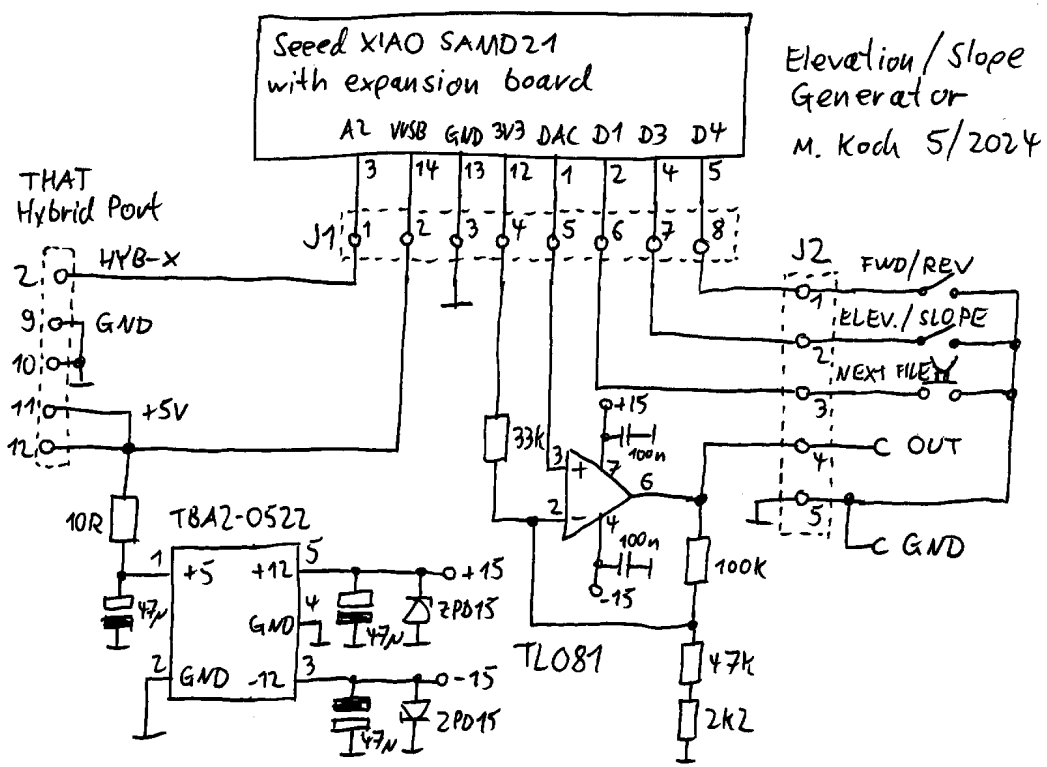
4.5.9 Elevation / Slope Generator with SD Card and OLED Display

This is a generator for elevation or slope signals, usable for simulating a vehicle on a realistic track. The input signal is X from THATs hybrid port (0 to 1 machine unit), this is the position along the track. The output signal can be either elevation or slope, selectable by a switch.

The track can be created in the excellent "BRouter" website and exported as a *.kml file: https://brouter.de/brouter-web

Multiple *.kml files can be copied to a Micro SD card, which is then plugged into this generator. You can scroll through the files with a pushbutton. With another switch the direction of the track can be reversed. The scaling factors and the elevation or slope curve are shown in the OLED display.

This is the circuit:



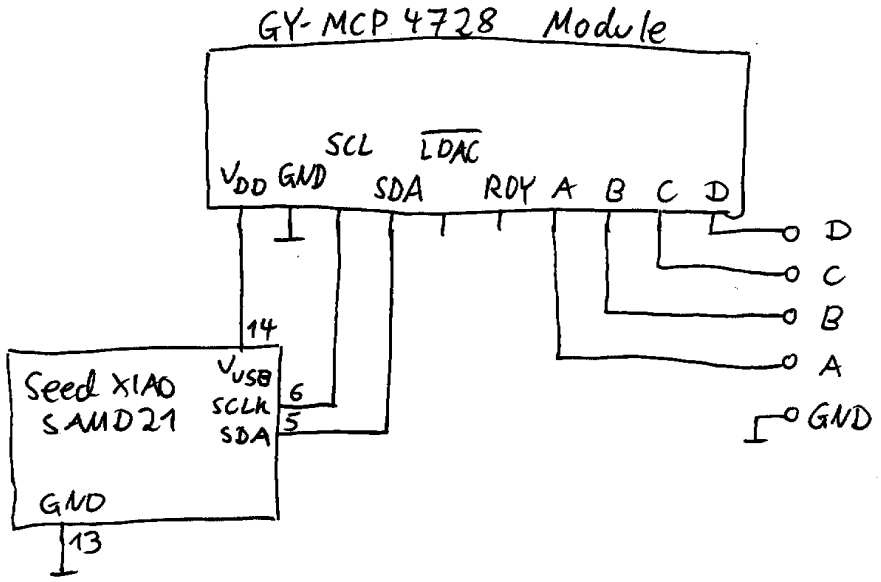
4.5.10 MCP4728 4-Channel DAC

MCP4728 Datasheet: https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/22187E.pdf

MCP4728 Modules are available very cheap on Aliexpress:
https://de.aliexpress.com/item/1005005797422191.html

For a sample program see also:
https://forum.electro-smith.com/t/lets-add-4-more-dacs-cvs-with-quad-dac-also-i2c-tips/3428/2

My test circuit:



It should be noted that the GY-MPC4728 module is used here with 5V supply voltage, so that the analog output valtages are in the 0 to 5 V range if VDD is used as reference, or 0 to 4.095 V if the internal reference is used. However the logic signals SDA and SCLK are 3.3V level and the SAMD21 microcontroller isn't 5 V compatible. It's no problem in this case because the SDA output of the MCP4728 is open-collector with a 8.2 kΩ pullup resistor to 5 V, resulting in 0.6 mA flowing into the input clamp diode of the microcontroller, which is acceptable.

Sample program:

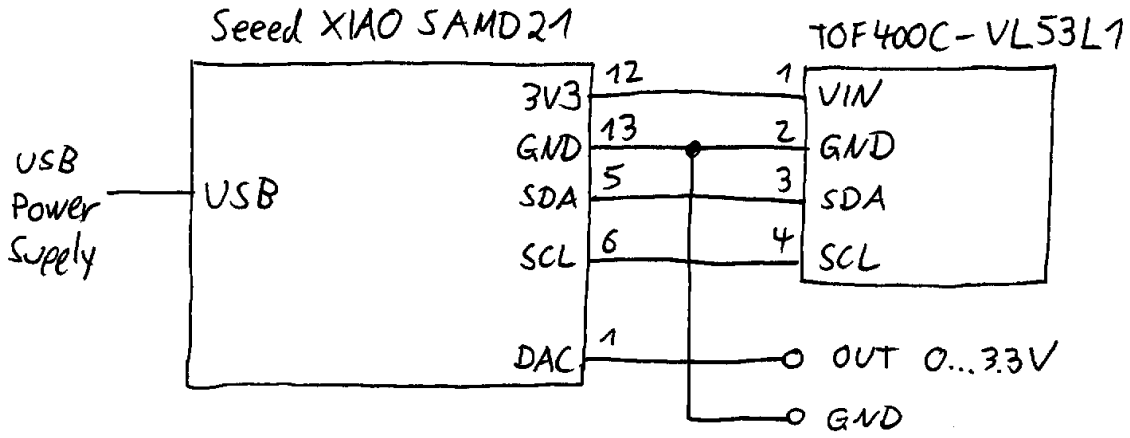
```
#include <Adafruit_MCP4728.h>

Adafruit_MCP4728 mcp;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10);
  Serial.println("Adafruit MCP4728 test!");
  if (!mcp.begin())
  {
    Serial.println("Failed to find MCP4728 chip");
    while (1);
  }
}

void loop()
{
  mcp.setChannelValue(MCP4728_CHANNEL_A, 4000, MCP4728_VREF_INTERNAL, MCP4728_GAIN_2X);   // 4.0 V with internal reference
  mcp.setChannelValue(MCP4728_CHANNEL_B, 4000, MCP4728_VREF_INTERNAL, MCP4728_GAIN_1X);   // 2.0 V with internal reference
  mcp.setChannelValue(MCP4728_CHANNEL_C, 2048);                                           // VDD/2 with VDD as reference
  mcp.setChannelValue(MCP4728_CHANNEL_D, 1024);                                           // VDD/4 with VDD as reference
  delay(100);
}
```

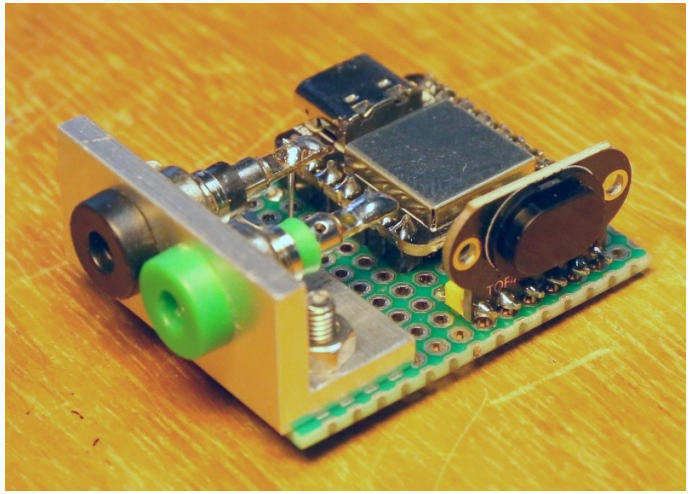
Note: There is a bug in the "Wire" library (which is included in the "Adafruit_MCP4728" library). If the I2C device isn't connected to the SDA and SCLK pins at all (these pins of the microcontroller are open), then the library call to wire.begin() doesn't return any error message. It looks like a successful connection.

Data written to the I2C port, add 0x1000 to value for "Gain_x2" and add 0x8000 for internal reference:

Byte 1 Write to adress 0x60	Byte 2 Channel A, B, C, D	Byte 3 High-Byte of value	Byte 4 Low-Byte of value
0x60	0x40	0x9F	0xA0
0x60	0x42	0x8F	0xA0
0x60	0x44	0x08	0x00
0x60	0x46	0x04	0x00

4.5.11 Time of Flight Sensor





If you are using version 1.6.2 or later of the Arduino software (IDE), you can use the Library Manager to install this library:

1. **In the Arduino IDE, open the "Sketch" menu, select "Include Library", then "Manage Libraries...".**
2. **Search for "VL53L1X".**
3. **Click the VL53L1X entry in the list.**
4. **Click "Install".**

Software:

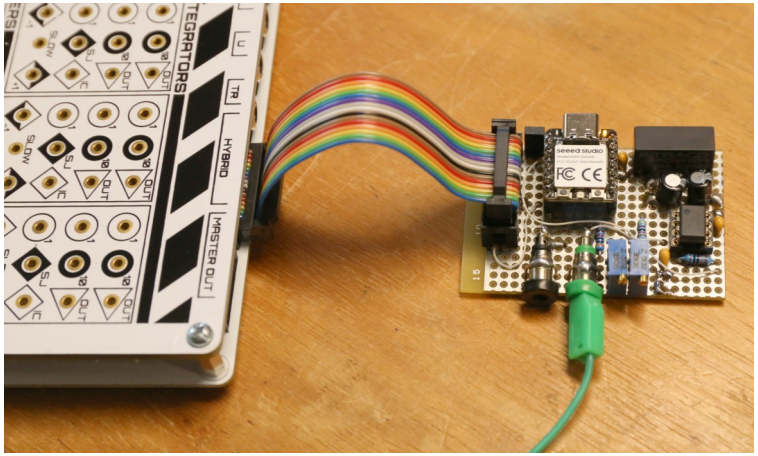
```
/*
This example shows how to take simple range measurements with the VL53L1X. The
range readings are in units of mm.
*/

#include <Wire.h>
#include <VL53L1X.h>

VL53L1X sensor;
int x;

void setup()
{
```

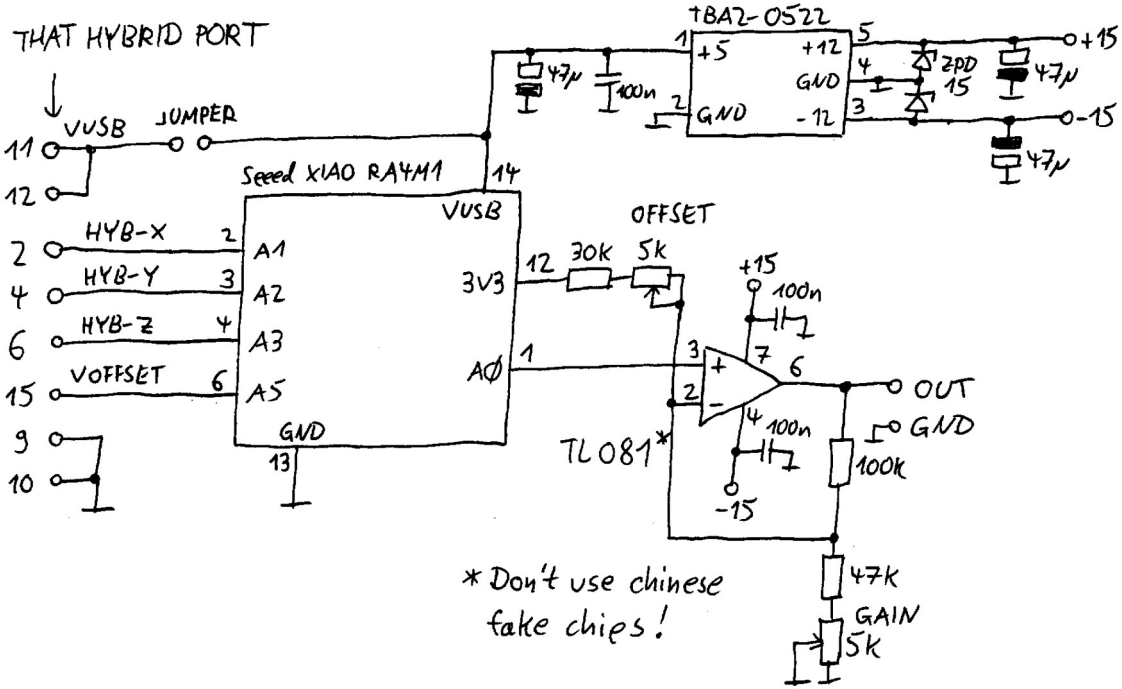
```
  while (!Serial) {}
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(400000); // use 400 kHz I2C

  sensor.setTimeout(500);
  if (!sensor.init())
  {
    Serial.println("Failed to detect and initialize sensor!");
    while (1);
  }

  // Use long distance mode and allow up to 50000 us (50 ms) for a measurement.
  // You can change these settings to adjust the performance of the sensor, but
  // the minimum timing budget is 20 ms for short distance mode and 33 ms for
  // medium and long distance modes. See the VL53L1X datasheet for more
  // information on range and timing limits.
  sensor.setDistanceMode(VL53L1X::Long);
  sensor.setMeasurementTimingBudget(33000);

  // Start continuous readings at a rate of one measurement every 50 ms (the
  // inter-measurement period). This period should be at least as long as the
  // timing budget.
  sensor.startContinuous(33);
}

void loop()
{
  x = sensor.read();
  if(x > 1023)
    x = 1023;
  analogWrite(A0,x);   // full scale is 1023mm
  Serial.print(x);
  if (sensor.timeoutOccurred()) { Serial.print(" TIMEOUT"); }
  Serial.println();
}
```

4.6 Seeed XIAO RA4M1

This microcontroller is similar to the Seeed XIAO SAMD21, but has 14-bit ADCs, a 12-bit DAC and a floating point unit.

4.6.1 Simple Function Generator





This is a generator for functions of up to three variables X, Y and Z from the hybrid port. The ADCs do calibrate themselves with the V_OFFSET voltage from the hybrid port. The output gain and offset must be adjusted with potentiometers.

My first tests with the RA4M1 module are disappointing. There is so much noise in the ADC measurements. The older SAMD21 module is much better (in the same circuit).

Software:

```
/*
Programmable Function Generator for THAT's Hybrid Port with Seeed XIAO RA4M1
Michael Koch, September 2024
*/

float x,y,z,f;
int v_offset;
float scaling;

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(D7, OUTPUT);   // This is a test pin for measuring the sample rate

  analogReadResolution(14);    // 12-bit or 14-bit ADC
  analogWriteResolution(12);   // 12-bit DAC

  // Serial.begin(115200);
  // while (!Serial)
  //   delay(10);

  v_offset = 0;                     // make a precision measurement of the v_offset voltage
  for(int i = 0; i < 10000; i++)    //  from THAT's hybrid port, this voltage should be
    v_offset += analogRead(A5);     //  exactly 10V / 610 * 100 = 1.6393V
  v_offset /= 10000;                // v_offset is approximately 2035 with 12-bic ADC
                                    //  or 8139 with 14-bit ADC

  scaling = 1.6393 / v_offset;      // scaling is approximatly 0.0008 with 12-bit ADC
                                    //  or 0.0002 with 14-bit ADC

  // Serial.print("Offset: ");
  // Serial.println(v_offset);
  // Serial.print("Scaling: ");
  // Serial.println((float)scaling, 6);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(D7, HIGH);                // test pin = high
  x = scaling * ((int)analogRead(A1) - v_offset);   // x signal in machine units
```

```
  // y = (float)(scaling * ((int)analogRead(A2) - v_offset));   // y signal in machine units
  // z = (float)(scaling * ((int)analogRead(A3) - v_offset));   // z signal in machine units
  digitalWrite(D7, LOW);                 // test pin = low

  // enter the user-defined function here, all variables are in machine units
  f = x;
  if ((x >= -.5) && (x < -.1)) f = -1;
  if ((x >= -.1) && (x < .1)) f = 0;
  if ((x >= .1) && (x < .5)) f = 1;

  if (f < -1) f = -1;                            // clip to -1
  if (f > 1) f = 1;                              // clip to +1
  analogWrite(A0,(int)(2047.5 + f * 2047.5));   // write f to the 12-bit DAC
}
```

Next test, trying to use an external reference voltage.

Schematic of the module: https://files.seeedstudio.com/wiki/XIAO-R4AM1/res/XIAO-RA4M1_SCH_PDF_v1.0_240719.pdf

µC Pin 43 = AVSS0 = connected to GND
µC Pin 42 = AVCC0 = connected to 3V3 via 0R (R4) and 1µF
µC Pin 45 = VREFH0 = P010 = connected to 3V3 via 0R (R5) and 100nF
µC Pin 44 = VREFL0 = P011 = connected to User LED
µC Pin 41 = VREFH = P012 = SDA = TP13 (marked "014")
µC Pin 40 = VREFL = P013 = SCL = TP14 (marked "013")

Analog power supply	AVCC0	Input	Analog voltage supply pin
	AVSS0	Input	Analog voltage supply ground pin
	VREFH0	Input	Analog reference voltage supply pin
	VREFL0	Input	Reference power supply ground pin
	VREFH	Input	Analog reference voltage supply pin for D/A converter
	VREFL	Input	Analog reference ground pin for D/A converter

It's not possible to use an external reference voltage for the ADC (without modifying the RA4M1 module), because both AVCC0 and VREFH0 are connected to 3.3V via 0R resistors.

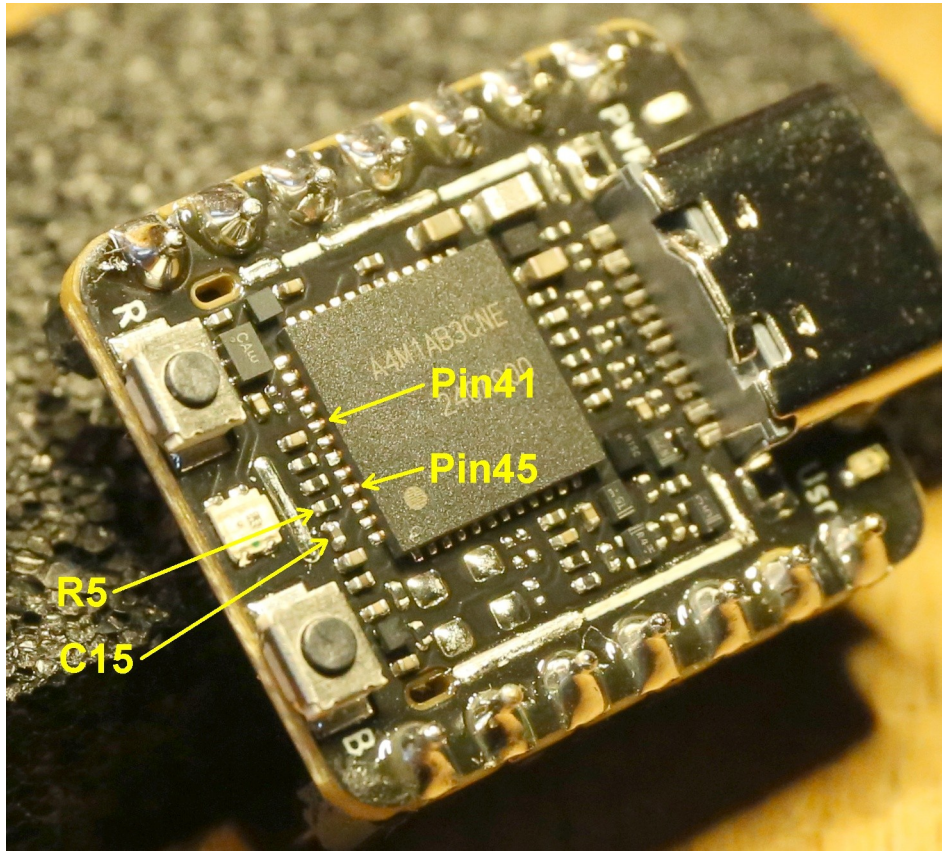
An external reference for the DAC can be connected to VREFH = P012 = TP13 and VREFL = P013 = TP14, but it seems this is not yet supported in the library.

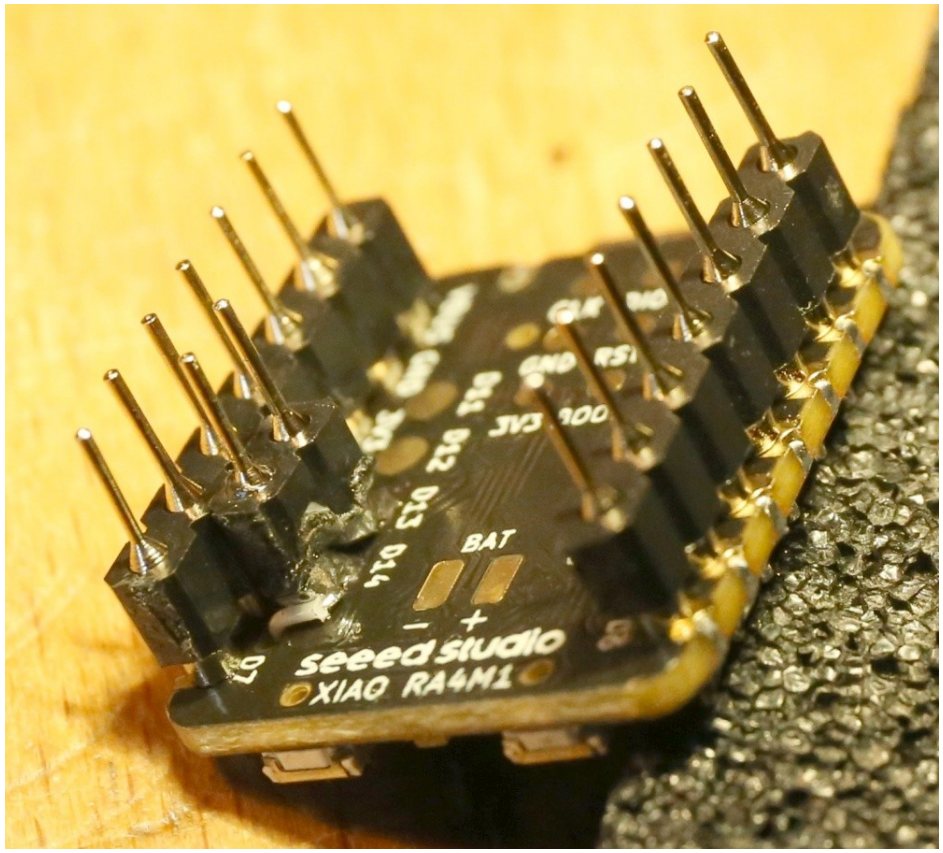
See page 1158 in the RA4M1 users manual:
https://www.renesas.com/us/en/document/mah/renesas-ra4m1-group-users-manual-hardware?r=1054146

Obviously the value 06h must be written to the D/A VREF Control Register (DAVREFCR). Before writing 06h, the value 00h should be written.

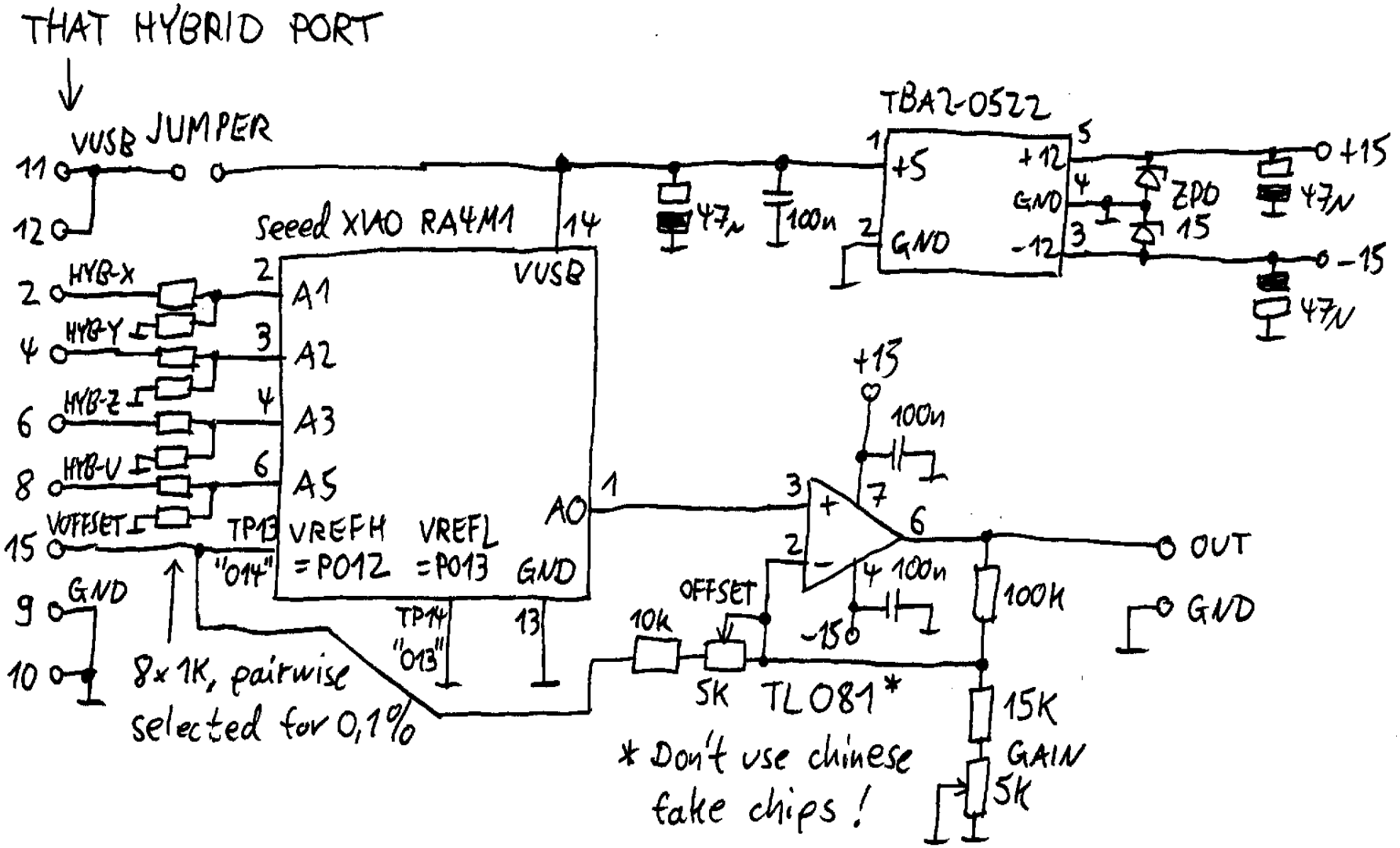
Address: DAC12.DAVREFCR = 4005E007h

How to modify the module for using an external reference voltage? The 0R resistor R5 must be removed, so that VREFH0 is disconnected from 3V3. Then pin 45 (or the upper end of C15) must be connected to REFH at pin 41 (or to TP13 on the lower side, which is marked "014"). Connect VREFL (marked "013") to GND and VREFH (marked "014") to the 1.6393V reference voltage. It's difficult to solder these two pins because the solder pads are off-grid.





Schematic:


THAT HYBRID PORT
VUSB JUMPER
11
12
TBA2-0522
+5
GND
+12
GND
-12
ZDD 15
+15
-15
47n
100n
Seeed XIAO RA4M1
VUSB
A1
A2
A3
A5
A0
TP13 "014" = P012
VREFH = P013
VREFL
GND
TP14 "013"
HYB-X
HYB-Y
HYB-Z
HYB-V
VOFFSET
GND
8x 1K, pairwise selected for 0,1%
10k
5K
OFFSET
TL081*
100k
15K
GAIN
5K
OUT
GND
* Don't use chinese fake chips!

See also: https://renesas.github.io/fsp/group___a_d_c.html **BGR = Band Gap Reference, internal reference voltage ?**

enum adc_vref_control_t

Enumerator	
ADC_VREF_CONTROL_VREFH	VREFAMPCNT reset value. VREFADC Output voltage is Hi-Z.
ADC_VREF_CONTROL_1_5V_OUTPUT	BGR turn ON. VREFADC Output voltage is 1.5 V.
ADC_VREF_CONTROL_2_0V_OUTPUT	BGR turn ON. VREFADC Output voltage is 2.0 V.
ADC_VREF_CONTROL_2_5V_OUTPUT	BGR turn ON. VREFADC Output voltage is 2.5 V.
ADC_VREF_CONTROL_AVCC0_AVSS0	High potential is AVCC0, low potential is AVSS0.
ADC_VREF_CONTROL_VREFH0_AVSS0	High potential is VREFH0, low potential is AVSS0.
ADC_VREF_CONTROL_IVREF_AVSS0	High potential is internal reference voltage, low potential is AVSS0. When the high potential is set to the internal reference voltage, wait 5 us after R_ADC_Open() to start an ADC measurement.
ADC_VREF_CONTROL_AVCC0_VREFL0	High potential is AVCC0, low potential is VREFL0.
ADC_VREF_CONTROL_VREFH0_VREFL0	High potential is VREFH0, low potential is VREFL0.
ADC_VREF_CONTROL_IVREF_VREFL0	High potential is internal reference voltage, low potential is VREFL0. When the high potential is set to the internal reference voltage, wait 5 us after R_ADC_Open() to start an ADC measurement.

Software:

```
//  Programmable Function Generator for THAT's Hybrid Port with Seeed XIAO RA4M1
//  Michael Koch, September 2024

float x,y,z,u,f;
int offset;
float scaling;
volatile uint8_t *DAVREFCR = (volatile uint8_t*)0x4005E007;

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(D7, OUTPUT);    // This is a test pin for measuring the sample rate
  analogReference(AR_EXTERNAL);    // External reference for ADC
  analogReadResolution(12);        // 12-bit or 14-bit ADC
  analogWriteResolution(12);       // 12-bit DAC

  // The v_offset voltage from THAT's hybrid port is exactly 10V / 610 * 100 = 1.6393V
  // This voltage is used as reference for the ADC
```

```
  // Signal voltage at the hybrid port X, Y, Z, U:   0.6393V ... 1.6393V ... 2.6393V
  // Signal voltage at ADC inputs A2, A3, A5:        0.3196V ... 0.8196V ... 1.3196V
  // ADC value (12-bit):                               799   ...  2048   ...  3297
  // ADC value (14-bit):                              3194   ...  8192   ... 13189

  offset = 2048;                  // for 12-bit ADC
  // offset = 8192;               // for 14-bit ADC
  scaling = 1.6393 / offset;      // scaling is approximatly 0.0008 with 12-bit ADC
                                  //  or 0.0002 with 14-bit ADC
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(D7, HIGH);               // test pin = high
  x = scaling * ((int)analogRead(A1) - offset);   // x signal in machine units
  // y = scaling * ((int)analogRead(A2) - offset);   // y signal in machine units
  // z = scaling * ((int)analogRead(A3) - offset);   // z signal in machine units
  // u = scaling * ((int)analogRead(A5) - offset);   // u signal in machine units
  digitalWrite(D7, LOW);                // test pin = low

  // enter the user-defined function here, all variables are in machine units
  f = x;

  if (f < -1) f = -1;                           // clip to -1
  if (f > 1) f = 1;                             // clip to +1
  //analogWrite(A0,(int)(511.5 + f * 511.5));   // write f to the 10-bit DAC
  analogWrite(A0,(int)(2485 + f * 1610));       // write f to the 12-bit DAC, 0.35V to 1.6393V
  *DAVREFCR = 0;                                // prepare for switching
  *DAVREFCR = 6;                                // use external reference for DAC
}
```

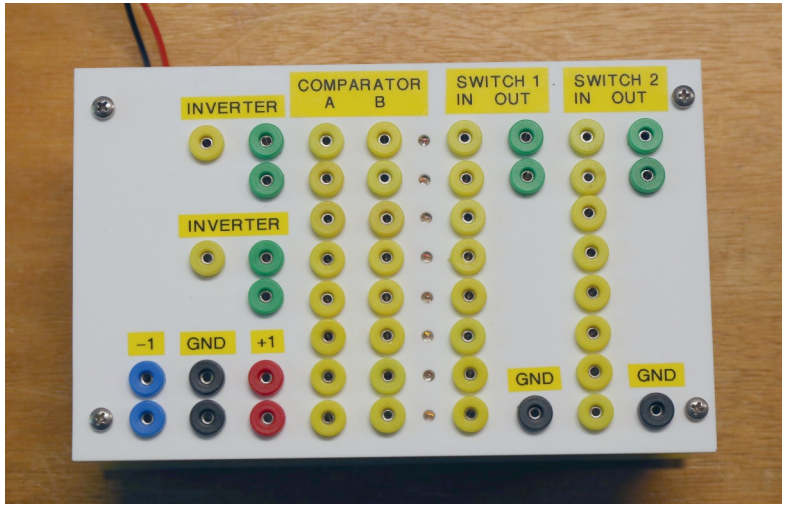
The measurement results with this modified RA4M1 module aren't perfect, but better as with the default 3.3V reference.

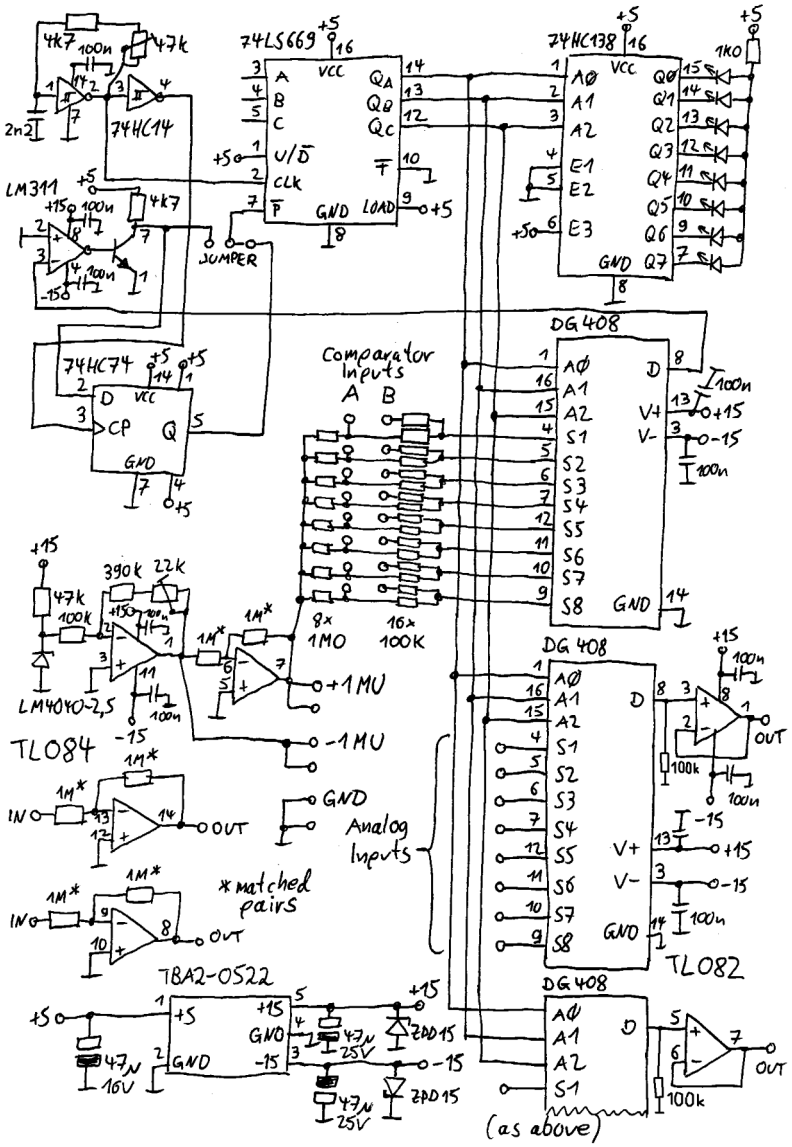
Update in October 2024: There was too much noise on the 3.3 V supply for the ADC. After changing R4 from 0Ω to 15Ω the noise should be much less. For details see: https://forum.seeedstudio.com/t/ra4m1-huge-noise-in-adc-values/281591/5

4.7 Counter / Analog Multiplexer

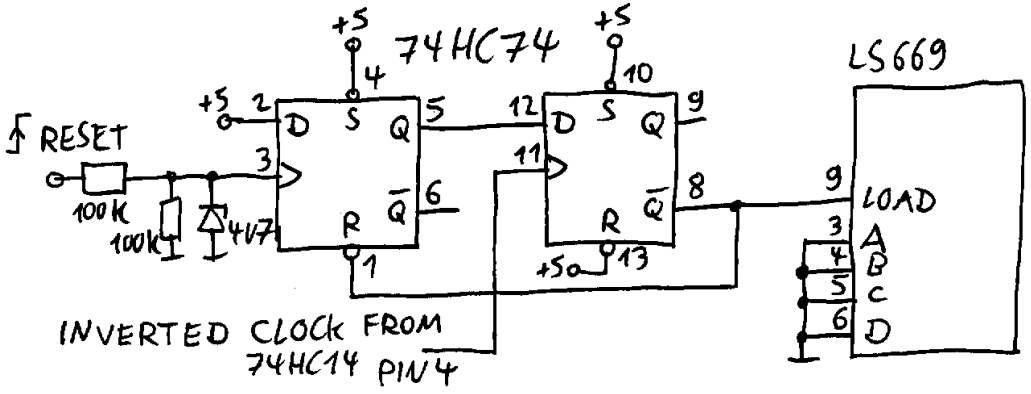
What does the circuit do? In the top left corner is a clock generator, running at about 20 kHz. This is the clock signal for a 3-bit counter 74LS669 (could possibly be replaced by 74HC191 or 74HC193 which have different pinouts, not yet tested). The counter has 8 possible states, which are shown with 8 LEDs at the output of the 74HC138.
The 3-bit bus drives three DG408 analog switches. Each of these chips has 8 analog inputs and one analog output. The first switch is used to generate the count enable signal. Each state has its own pair of compare inputs A and B. If A+B>0, then the counter advances to the next state. The A inputs have a positive level by default, if no signal is connected to them. That means the counter does count very fast through those states that are unused.
The other two analog switches have buffered outputs.
I did also add two inverters and a circuit for -1 and +1 machine unit signals.

The clock frequency is adjustable from about 10 kHz to 100 kHz. I found that 20 kHz is a safe value, so that when the comparator condition is met and the analog switch advances to the next state, the comparator output signal must be less than zero again before the next rising edge of the clock signal. Otherwise the counter would advance two or more steps.

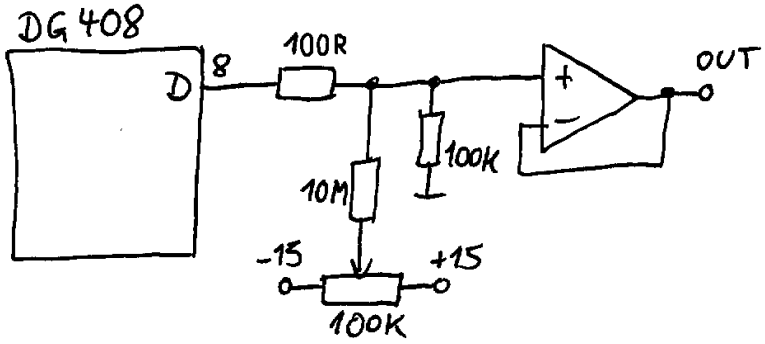




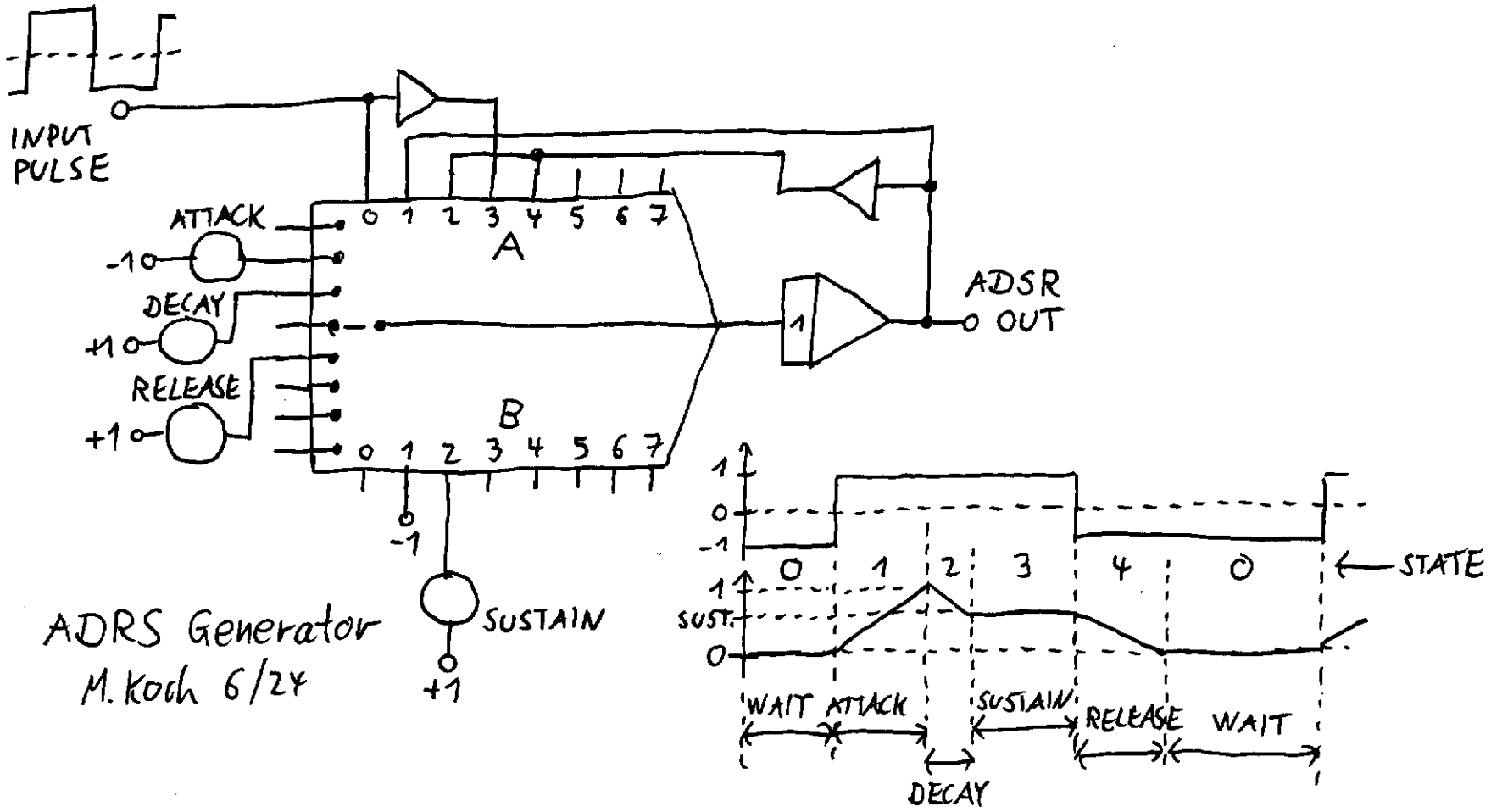
Later I added this circuit, which resets the counter to state 0 at the rising edge of the reset signal.



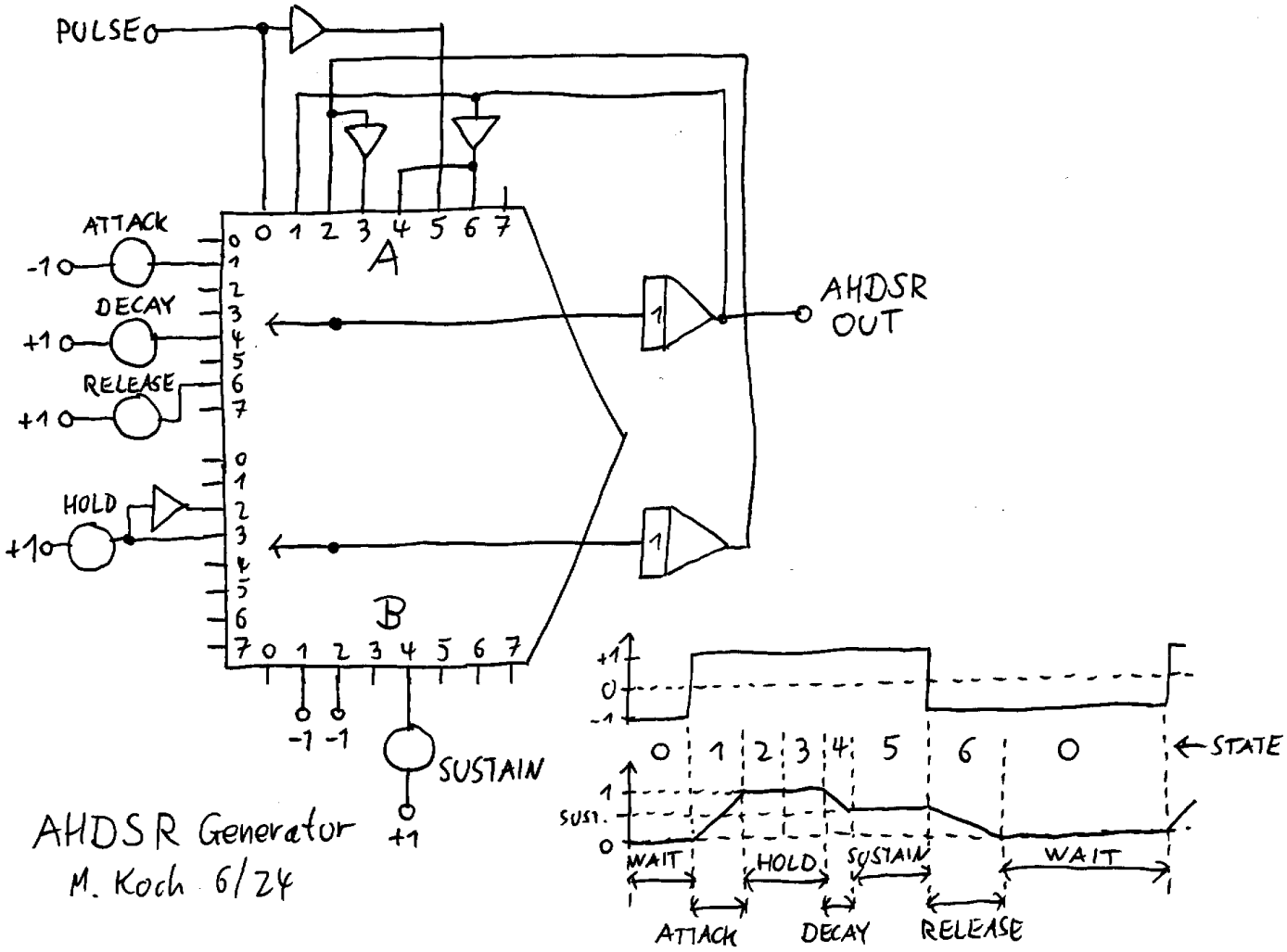
I also added this circuit for offset correction at the output amplifieres (for both channels):



4.7.1 ADSR envelope generator (Attack-Decay-Sustain-Release)

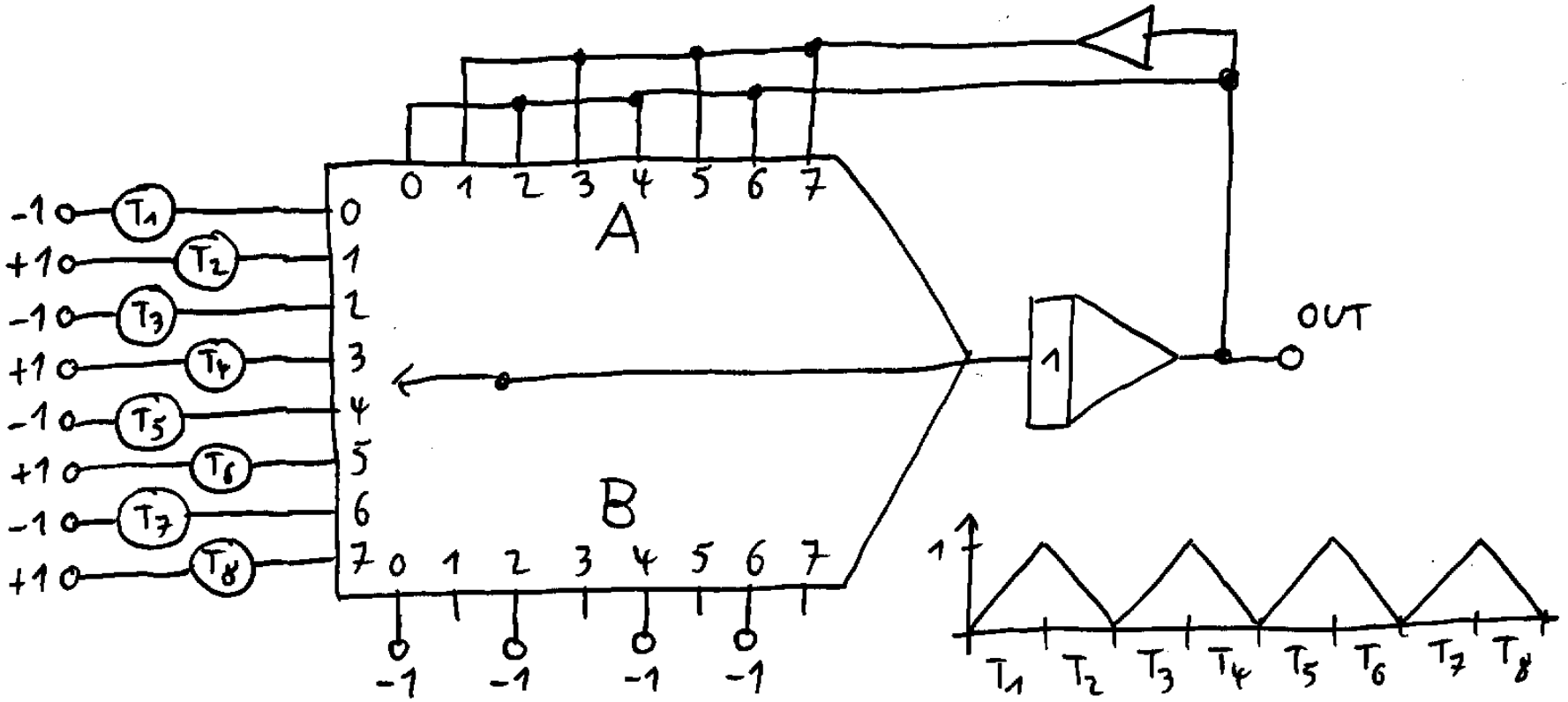


4.7.2 AHDSR envelope (Attack-Hold-Decay-Sustain-Release)



4.7.3 8-State-Sequencer

With this circuit the counter cycles through all 8 states with adjustable durations. The second 8:1 switch can be used for designing many different signals.



4.7.4 Heart Sound Simulator

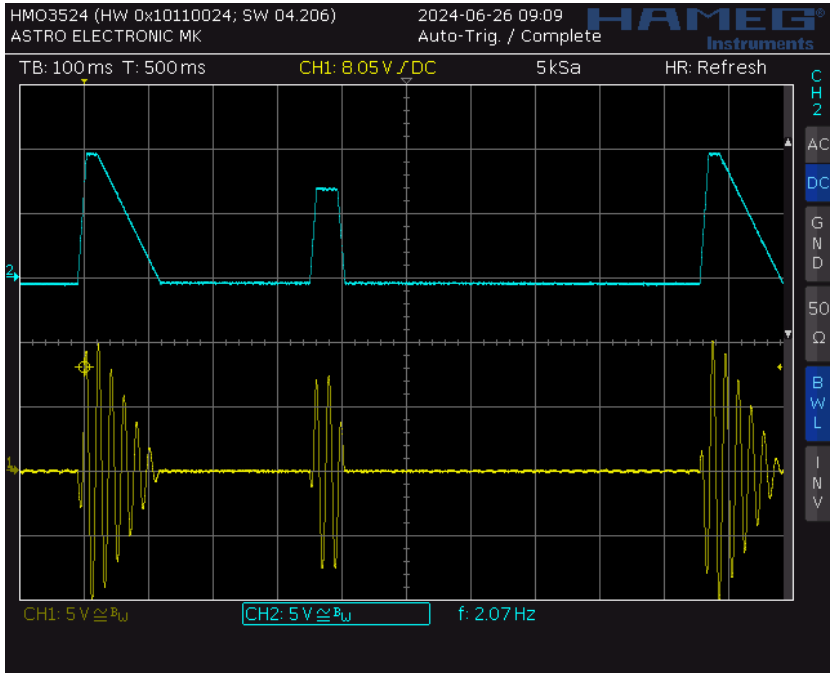
First I downloaded two heart sound files from Wikipedia:

https://en.wikipedia.org/wiki/Heart_sounds

https://de.wikipedia.org/wiki/Herzton

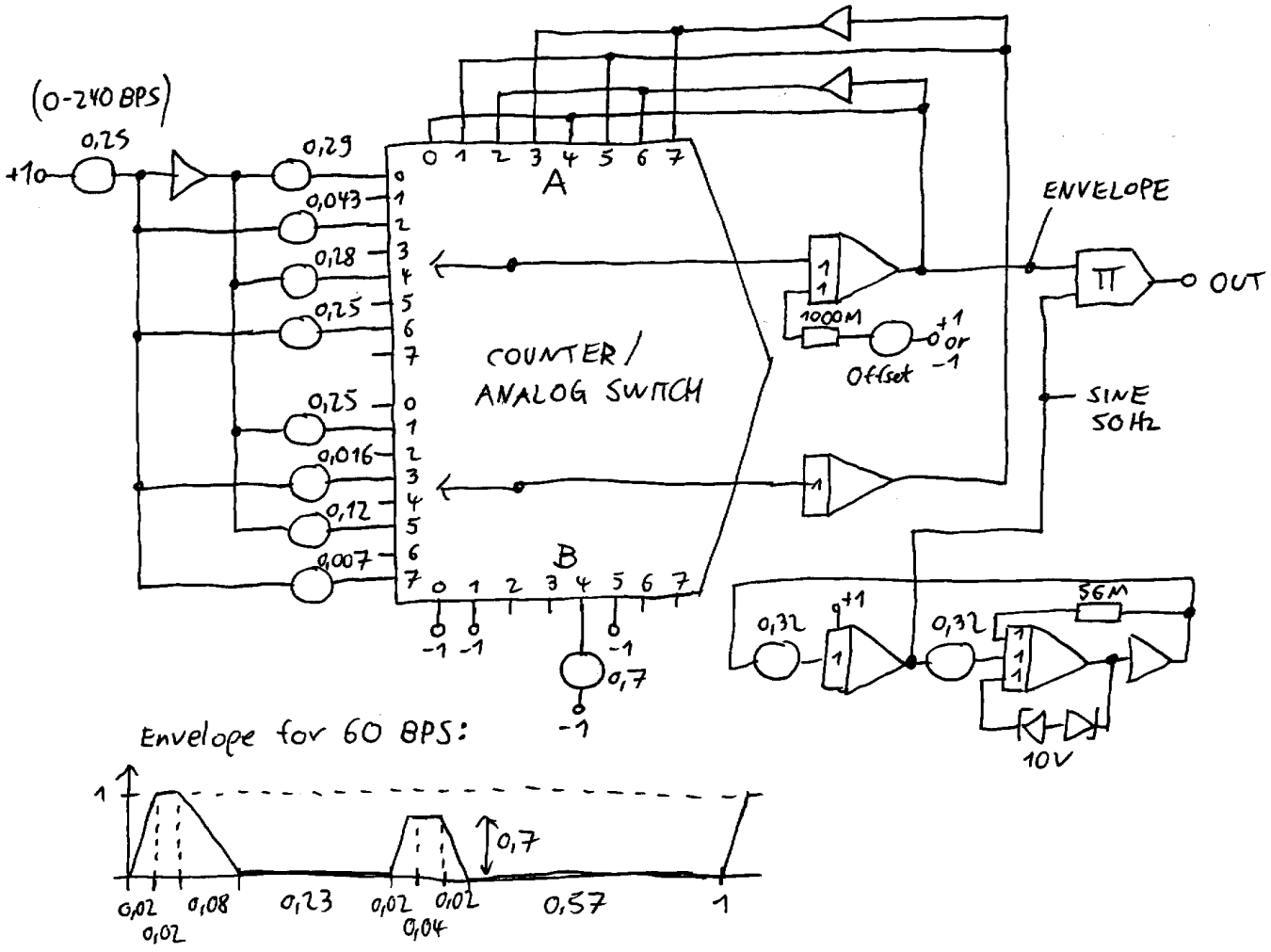
Then I analyzed these files with Sonic Visualiser, which is a nice free software for visualising the waveform of audio files:
https://www.sonicvisualiser.org

Then I drew the envelope curve and simulated it with the counter / analog switch. The modulation frequency is typically in the 40 to 80 Hz range, with best results between 40 and 50 Hz.



Note 1: Some of the coefficients are very small. It would be better to replace them by two coefficients in series.

Note 2: The offset correction at the input of the integrator was later built into the counter/analog switch circuit.

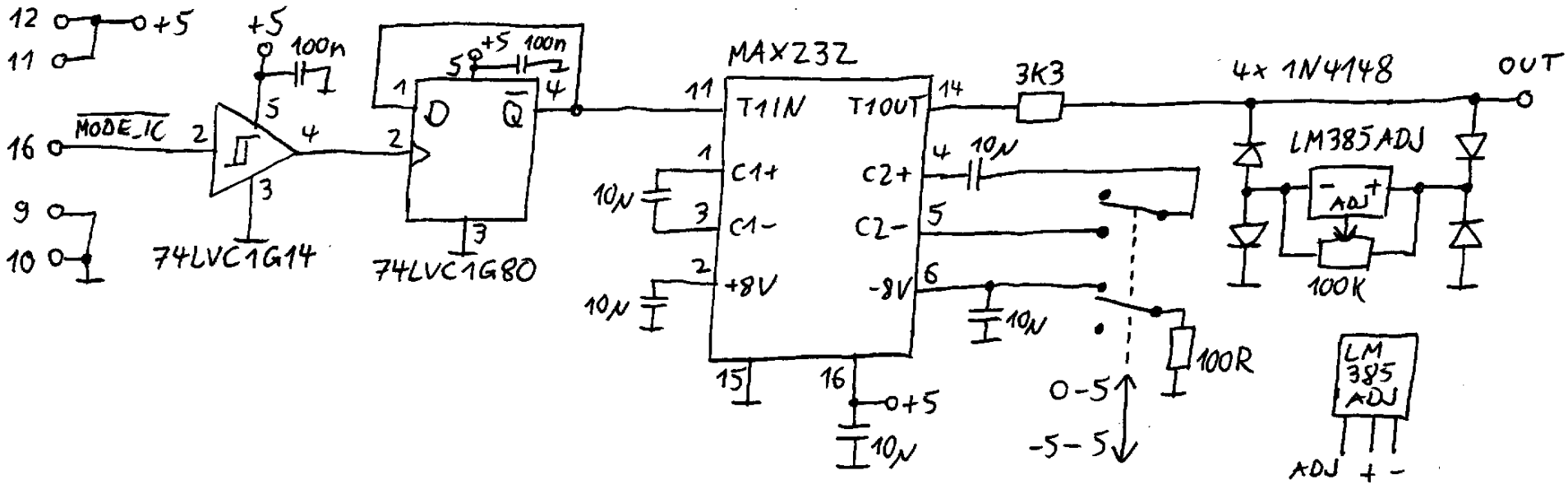


4.8 Flip-Flop at the Hybrid Port

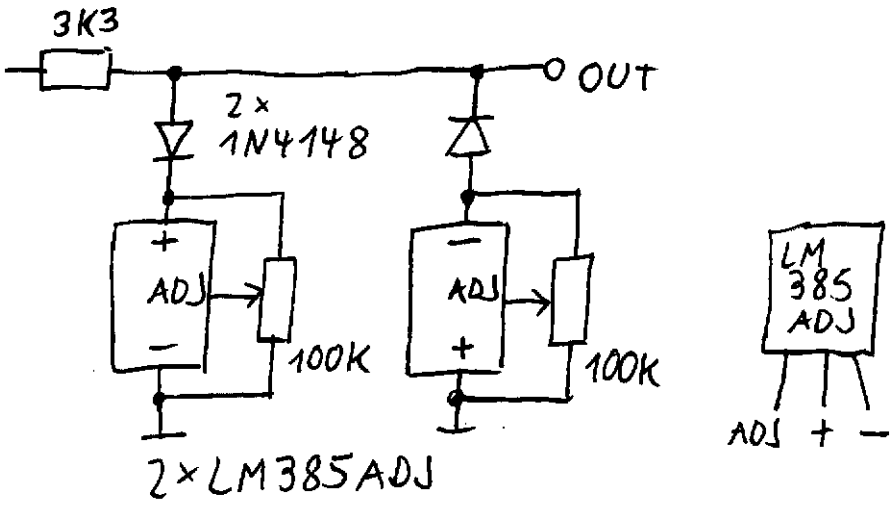
This circuit uses the /MODE_IC signal from the hybrid port to toggle a D-Flip-Flop. The output of the Flip-Flop alternates between low and high for consecutive THAT runs. It can be used for example in these cases:

1. **Calculate both branches of the Euler spiral. The output of the Flip-Flop controls a multiplexer, which selects one of the two branches.**
2. **Toggle between two different initial conditions, so that two different simulations are shown alternatingly on the screen.**

The schematic diagram has a few details that must be explained.
Why is a Schmitt Trigger required at the input of the Flip-Flop? Because the slewrate of the /MODE_IC signal is too slow. It didn't work without the Schmitt-Trigger.
If you need only an unstabilized 0V/5V signal, you can use the output of the Flip-Flop, and the right half of the circuit is unnecessary.
But I wanted a stabilized -5V/+5V signal or a 0V /+5V signal. The MAX232 is used as a level converter to +-8V. The negative -8V supply can be disabled with the switch. That makes the signal at pin 14 slow, but for this application it's still fast enough.
Finally the signal is stabilized to +-5V by an adjustable voltage reference within a bridge rectifier.



Later I modified the stabilization circuit with two LM385ADJ voltage references. This is better, because the forward voltage of the four 1N4148 diodes wasn't exactly equal. With the new circuit, the positive and negative levels can be adjusted separately:

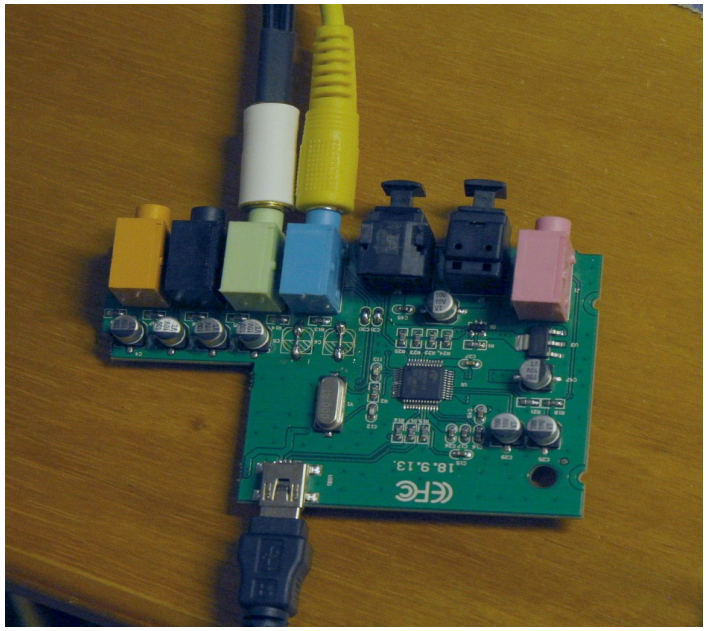


4.9 FFplay with USB Soundcard Input / Output

FFplay can be used for reading a signal from an audio input, processing the signal with an arbitrary transfer function, and sending the result to the computer's soundcard output. I did test it with a cheap USB soundcard. The audio inputs and outputs are AC coupled, but the capacitors can be removed and replaced by wires. Now the soundcard is DC coupled. The signal flow is as follows: A analog sawtooth signal goes into the soundcard input, it's converted to 16-bit with 8kHz sample rate, sent over USB to the PC, where the transfer function is applied in realtime, and then the signal is sent over the same USB cable back to the soundcard and converted to a voltage. I wanted to know the delay time from input to output. Unfortunately it's 80ms. Too slow for a realtime transfer function. This is the command line:

```
ffplay -nodisp -f dshow -audio_buffer_size 5 -sample_rate 8000 -sample_size 16 -channels 1 -i audio="Line (USB Sound
Device )" -af aeval='if(between(val(0),0,0.2),0,val(0))'
```

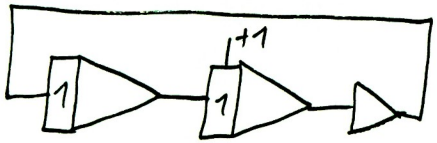
The last part of the command line is the transfer function. In words: If the input signal is between 0 and 0.2, then the output is 0, otherwise the output is the same as the input. On the oscilloscope magenta is input and yellow is output.

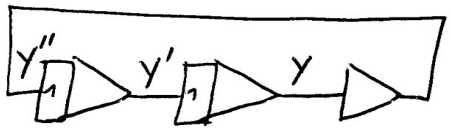


5 Signal Generators

5.1 Sine Generator

How can the frequency of this sine generator be calculated?





$$y = \sin(2\cdot\pi\cdot f\cdot t)$$

$$y' = \cos(2\cdot\pi\cdot f\cdot t)\cdot 2\cdot\pi\cdot f\cdot 1ms$$

$$y'' = -\sin(2\cdot\pi\cdot f\cdot t)\cdot(2\cdot\pi\cdot f)^2\cdot(1ms)^2$$

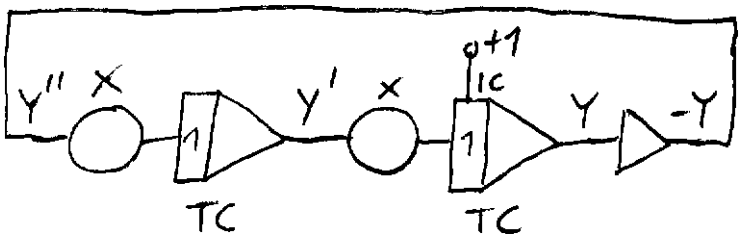
$$y = -y'' \Rightarrow (2\pi f)^2\cdot(1ms)^2 = 1$$

$$\Rightarrow 2\pi f\cdot 1ms = 1$$

$$\Rightarrow 2\pi f = 1kHz$$

$$\Rightarrow f = \frac{1kHz}{2\pi}$$

This is a sine oscillator with two coefficients for adjusting the frequency:



$\omega = 2 \cdot \pi \cdot f$

$y = \sin(\omega t)$

$y' = \cos(\omega t) \cdot \omega \cdot TC$

$y'' = -\sin(\omega t) \cdot \omega^2 \cdot TC^2$

$y = -y''$

$\sin(\omega t) = \sin(\omega t) \cdot \omega^2 \cdot TC^2$

$1 = \omega^2 \cdot TC^2$

$\omega = 1 / TC$

$f = 1 / (2 \cdot \pi \cdot TC)$

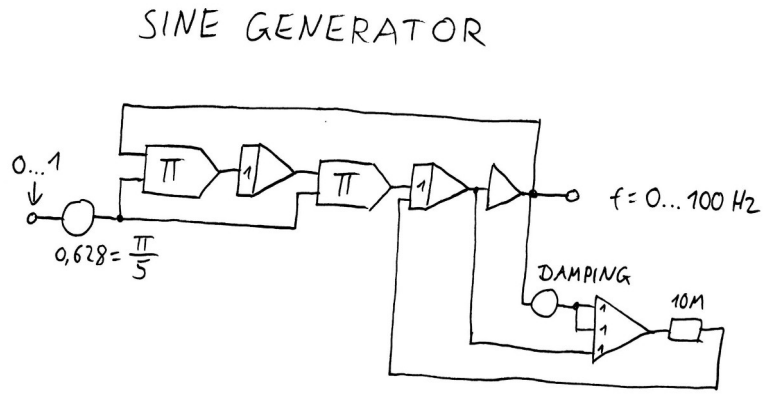
$TC = 1ms / x$

$f = x / (2 \cdot \pi \cdot 1ms)$

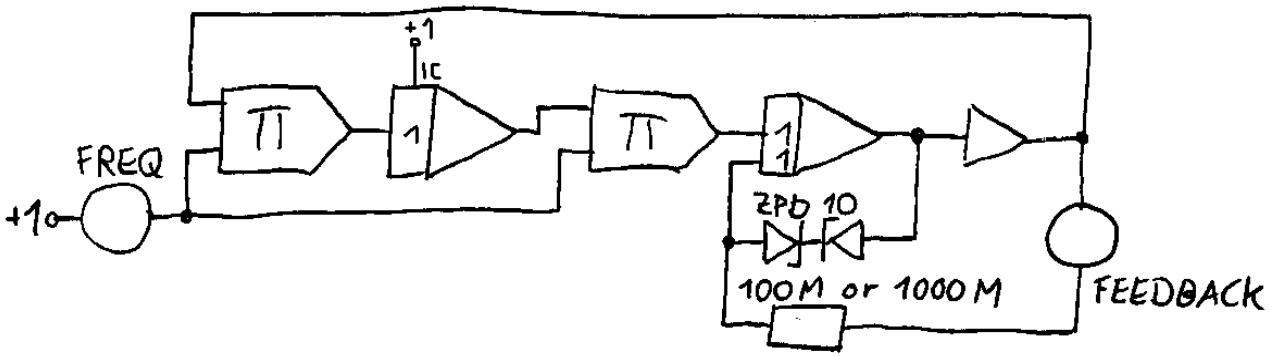
5.2 Voltage controlled Sine Generator

Forgot +1 initial condition at one of the integrators. The amplitude will slowly run away towards zero or overflow. This can be adjusted with the DAMPING coefficient. This is not amplitude stabilization. It does only allow for a longer time before the amplitude drifts away.

Integration Resistor	Integration Capacitor	Input = 0.628 (0.2·Π)		Input = 1	
		Frequency	Period	Frequency	Period
100 kΩ ("10" Input)	1nF (normal)	1 kHz	1 ms	1592 Hz (5000/Π) Hz	628 µs (200·Π) µs
1 MΩ ("1" Input)	1nF (normal)	100 Hz	10 ms	159 Hz (500/Π) Hz	6.28 ms (2·Π) ms
1 MΩ ("1" Input)	10nF (external)	10 Hz	100 ms	15.9 Hz (50/Π) Hz	62.8 ms (20·Π) ms
1 MΩ ("1" Input)	100nF (SLOW)	1 Hz	1 s	1.59 Hz (5/Π) Hz	628 ms (200·Π) ms
10 MΩ (external)	100nF (SLOW)	100 mHz	10 s	159 mHz (500/Π) mHz	6.28 s (2·Π) s
100 MΩ (external)	100nF (SLOW)	10 mHz	100 s	15.9 mHz (50/Π) mHz	62.8 s (20·Π) s
1 GΩ (external)	100nF (SLOW)	1 mHz	1000 s	1.59 mHz (5/Π) mHz	628 s (200·Π) s
1 GΩ (external)	1µF (with open amplifier)	100 µHz	10000 s	159 µHz (500/Π) µHz	6283 s (2000·Π) s
1 GΩ (external)	10µF (with open amplifier)	10 µHz	100000 s	15.9 µHz (50/Π) µHz	62832 s (20000·Π) s



Another circuit for amplitude stabilization:



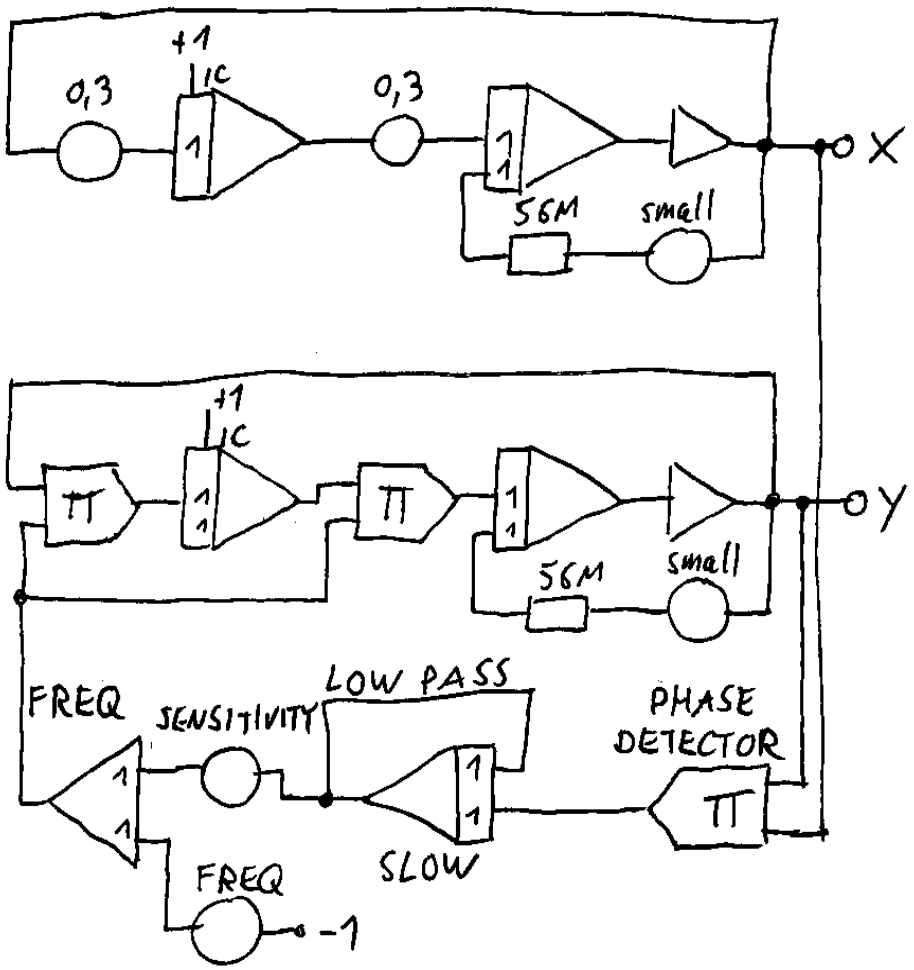
5.3 Phase Locked Loop

This circuit consists of two sine / cosine generators. The upper one has a fixed frequency and the lower one is adjustable by a control voltage.

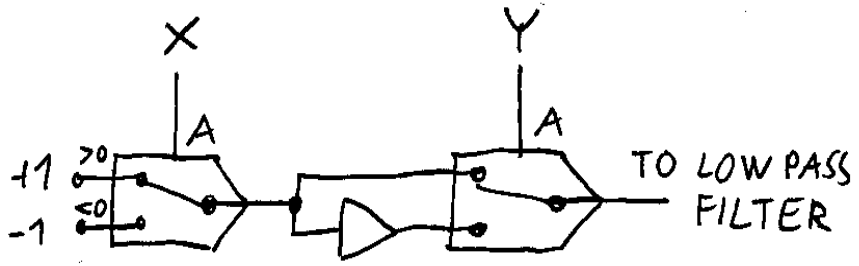
The question is how the two frequencies can be locked in an integer frequency ratio, for example 1:2.

A phase detector is required for a PLL (Phase Locked Loop). The simplest phase detector is an XOR gate, which is almost the same as a 4-quadrant multiplier. The output signal of the phase detector must be low-pass filtered, and then added to the control voltage.

Set SENSITIVITY to zero, tune the frequency to almost the correct frequency, then turn up SENSITIVITY. The second oscillator will lock to the first one. Look at the output signal of the low pass filter and then try to de-tune the frequency. Very interesting.

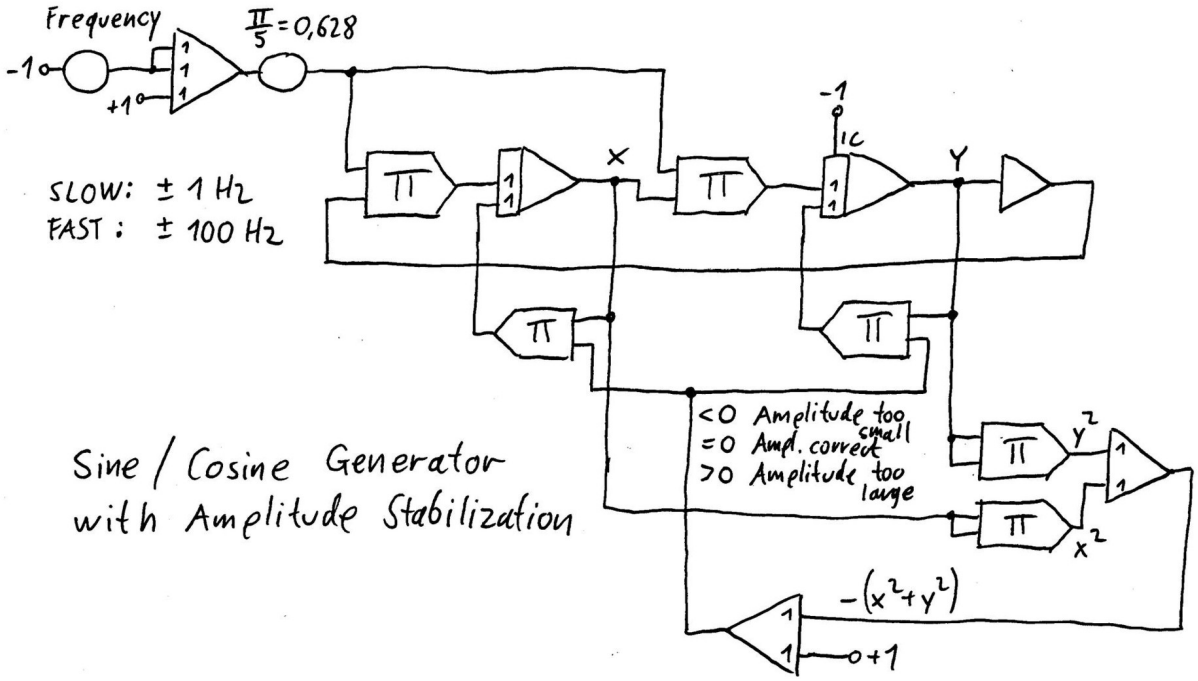


The phase detector can also be realized with two comparators. Then the whole circuit fits on one THAT.



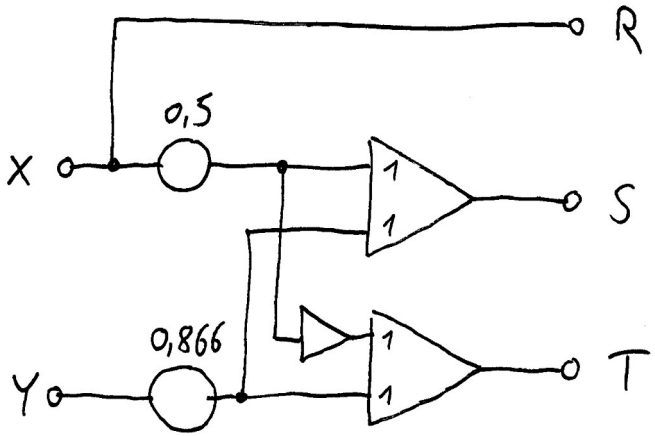
5.4 Sine / Cosine Generator with Amplitude Stabilization

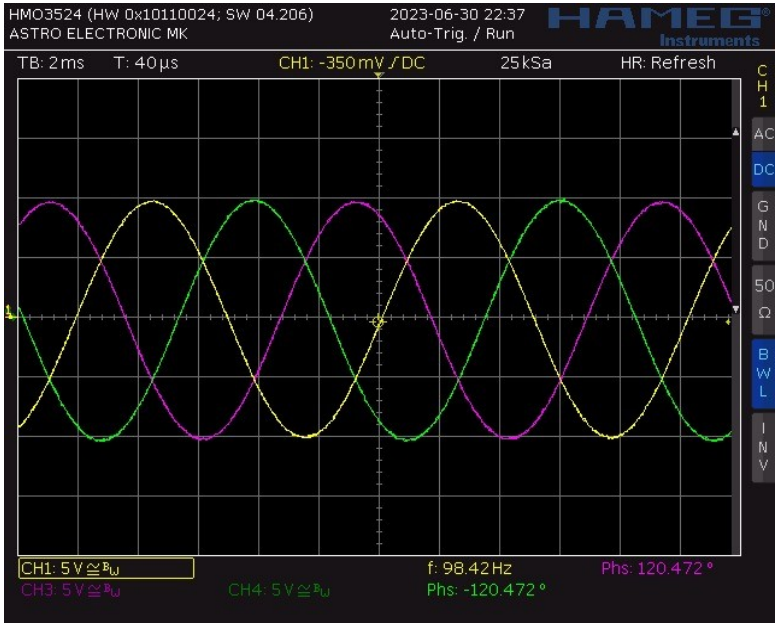
In analog sine / cosine generators the amplitude must somehow be stabilized. Two circuits with diodes are shown in Bernd's book "Analog and Hybrid Computer Programming" in figures 4.11 and 4.13. The drawback of amplitude stabilization with diodes is that it fails when the frequency becomes very small or zero.
In this circuit the stabilization does also work for small or zero frequency. The circuit in the lower right calculates the amplitude x^2+y^2 and compares it to the desired value 1. If an error is detected, it's corrected by the two multipliers in the center.
The drawback is that you need a total of 6 multipliers for this circuit.
This circuit can be further improved by making the amplitude corrections slower and allowing slightly larger error signals at the multiplier inputs, so that the multiplier's offset error doesn't produce extra errors. Insert two large resistors between the outputs of the error multipliers and the inputs of the integrators. 100 MΩ works fine, or even bigger resistor might be used.



5.5 3-Phase Generator

This circuit transforms the 90° phase shifted X and Y signals from the sine / cosine generator into three 120° phase shifted R, S, T signals. Works for all frequencies and for both turning directions.

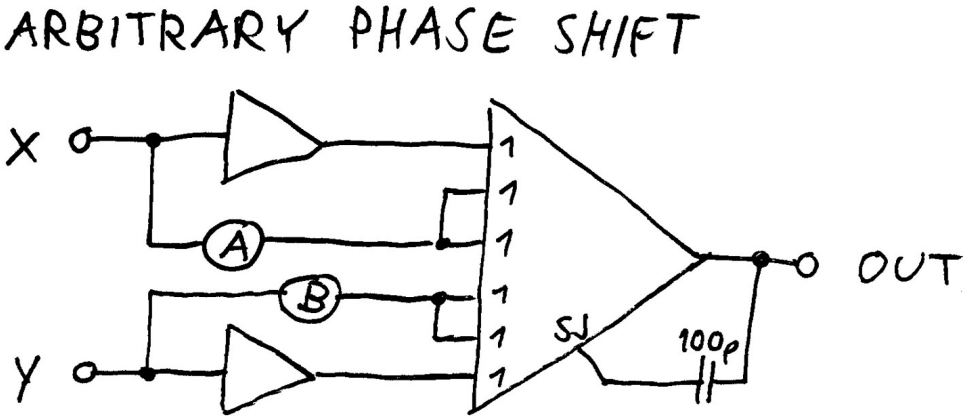




See also: Ammon, Werner: Schaltungen der Analogrechentechnik, Page 94: https://www.analogmuseum.org/library/Ammon_Schaltungen.pdf

5.6 Arbitrary Phase Shift

This circuit transforms the 90° phase shifted X and Y signals from the sine / cosine generator into an output signal with phi degrees phase shift with respect to the X signal. Works for all frequencies and for both turning directions. If you see too much noise in the output, slow down the summer with the 100pF capacitor.
Because this summer has 6 inputs, a XIR block is required. I found out that the cable from SJ(Summer) to SJ(XIR) is receiving interference from other cables. That's why I had to slow down the summer with the 100pF capacitor. It helps to keep the SJ to SJ cable as short as possible, and away from other cables. Then 33pF is sufficient.



$$A = 0{,}5\,(1 - \cos\varphi)$$
$$B = 0{,}5\,(1 - \sin\varphi)$$

5.7 Resolvers

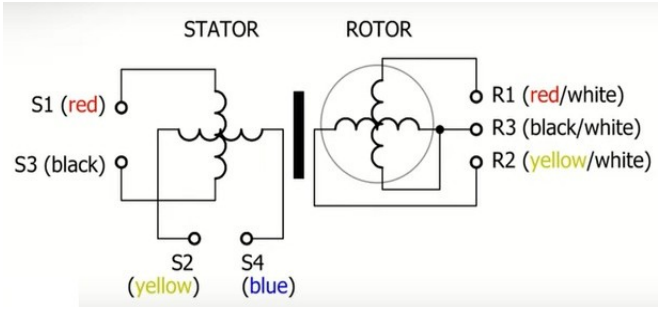
A resolver is an absolute angle encoder. The input and output signals are sine waves. https://en.wikipedia.org/wiki/Resolver_(electrical)

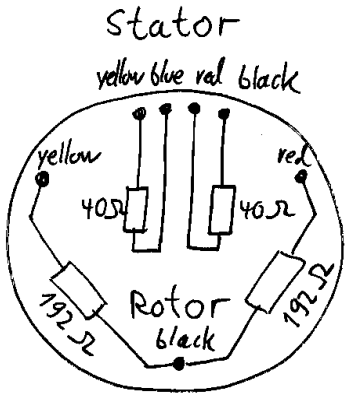
https://en.wikipedia.org/wiki/Synchro

I did use this resolver which I found on Ebay:

Microtecnica Autosyn Resolver AY 528 A 45 A 2, manufactured under Bendix licence. Rotor 2 phase 26V Stator 2 phase 11.8V

The resistance of the two stator coils is 40 Ω each. With a 150 Ω resistor in series the total resistance is 190 Ω, the current is 15.8 mA at 3V (0.3 machine units). The rotor has two 90° phase shifted coils with 192 Ω each. One end of them is connected together.





For more informations about resolvers and synchros, see also:

https://www.analog.com/en/education/education-library/synchro-resolver-conversion.html

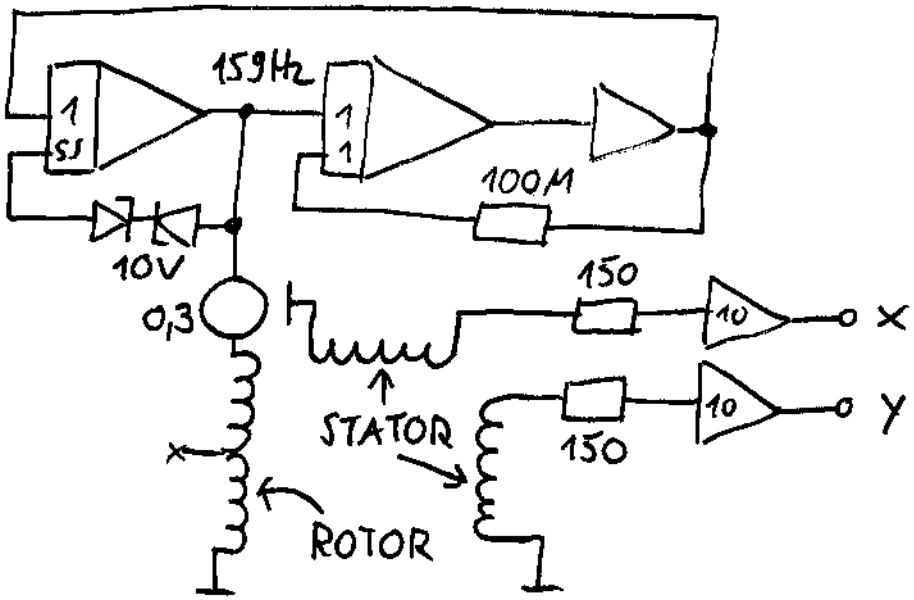
Resolvers have 2 phases with 90° phase shift, while synchros have 3 phases with 120° phase shift.

See also: https://analogmuseum.org/library/eai_handbook.pdf beginning at page 58

Nice video from Elektor: https://www.elektormagazine.com/news/resolver-synchro-introduction

Circuit 1:

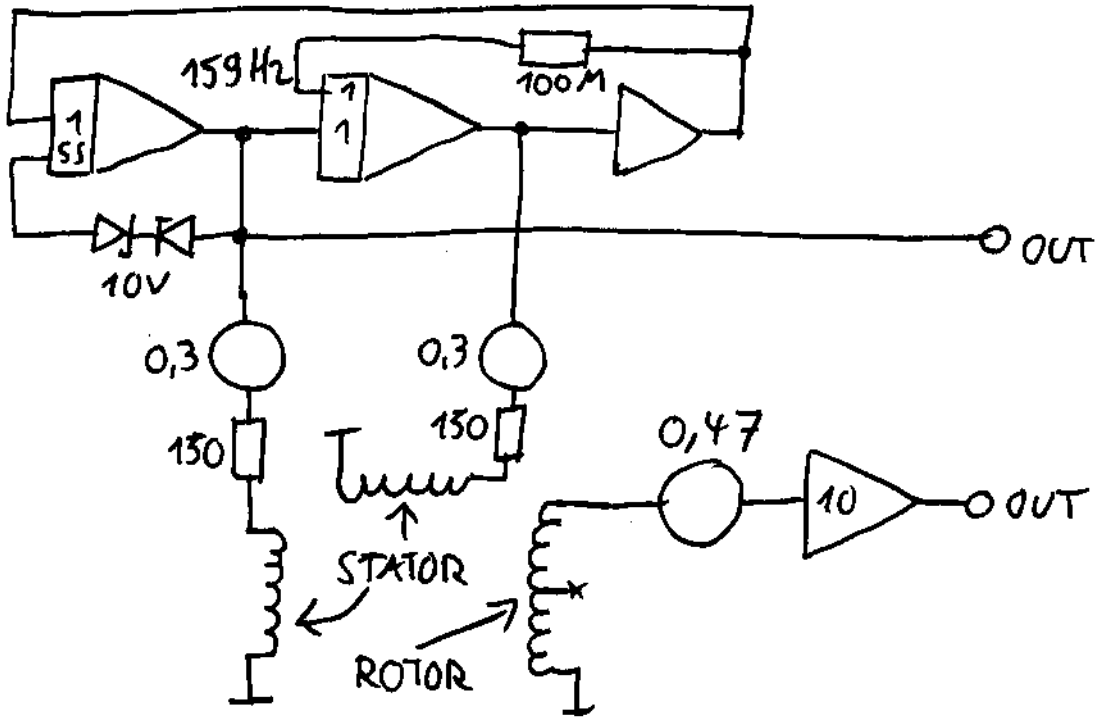
The input sine signal is connected to the rotor, and the voltages at the two stator coils are measured. The output signals are in phase and their ampliture and sign depends on the resolver's turning angle.



Alternatively it's also possible to connect the sine and cosine signals to the two rotor coils.

Circuit 2:

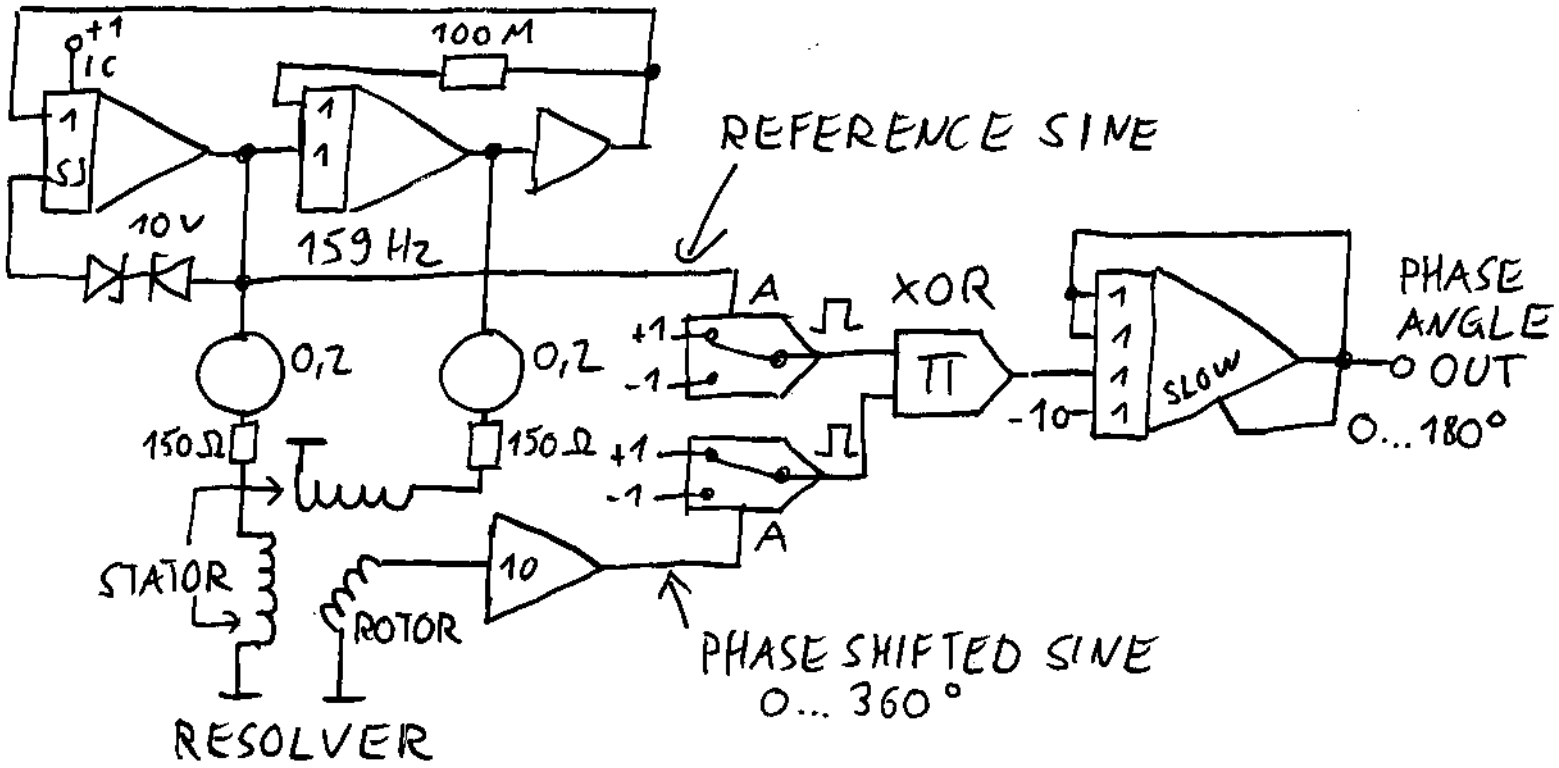
The input sine and cosine signals are connected to the stator coils, and the voltage at the rotor coils is measured. The output signal has constant amplitude and its phase shift with respect to the input signals depends on the resolver's turning angle.



Alternatively it's also possible to use both of the rotor coils, giving sine and cosine signals at the output.

5.8 Phase Angle Measurement

In this circuit the phase angle between two sine signals is measured. The reference signal is generated by a sine generator, and the phase shifted signal comes from a resolver. Both signals are converted to square waves and then a logical XOR is applied to them. If the logic levels are -1 and +1, the logical XOR can be realized with a multiplier. The result is shifted to the 0..1 range and low pass filtered.
Unfortunately the result is only from 0 to 180°, missing the sign of the phase shift. If you have an idea for +-180° output, please let me know.

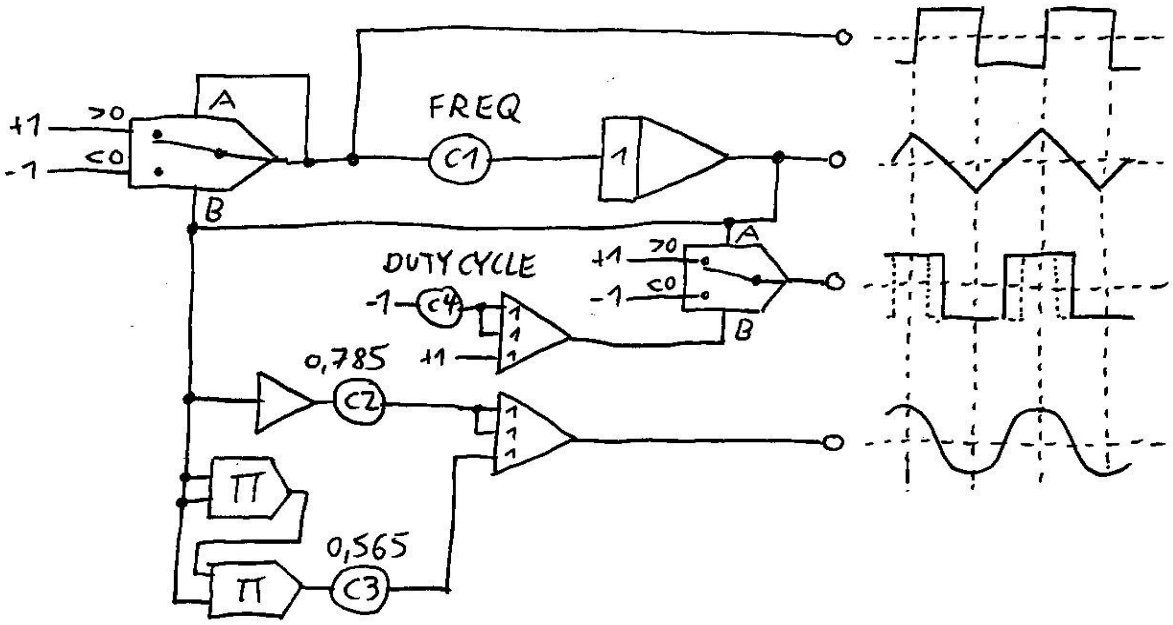


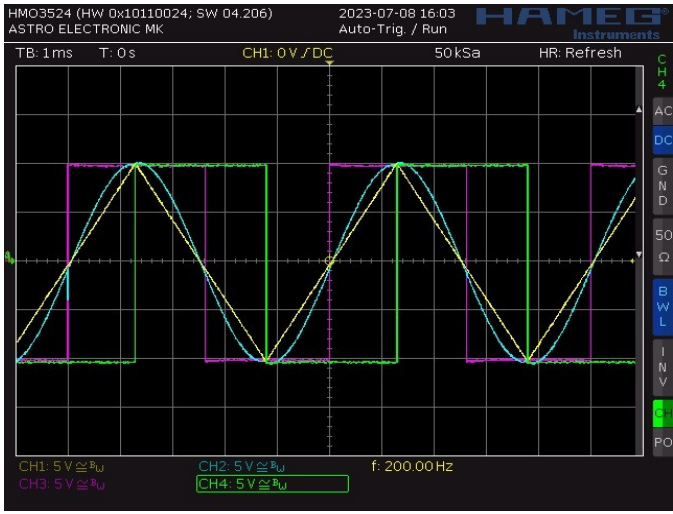
5.9 Approximates Sine and Square Wave with adjustable Duty Cycle

This circuit does simultaneously generate a square wave, a triangular wave, a 90° phase shifted square wave with adjustable duty cycle, and an approximated sine wave. For zero slope at the min/max points the coefficients are C2 = 0.75 and C3 = 0.5.

I did also try to calculate the C2/C3 ratio from the power series of the sin() function, but the result doesn't look good, probably because the higher orders are missing. The zero slope approach is better.

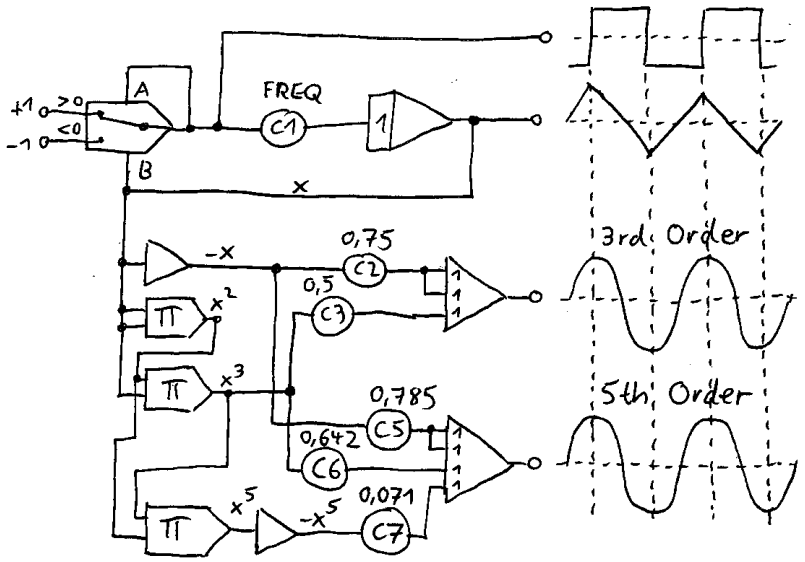
The frequency is f = 250Hz / (C1 · R), where R is the input resistor of the integrator in MΩ. The "1" input has 1 MΩ and the "10" input has 0.1MΩ.

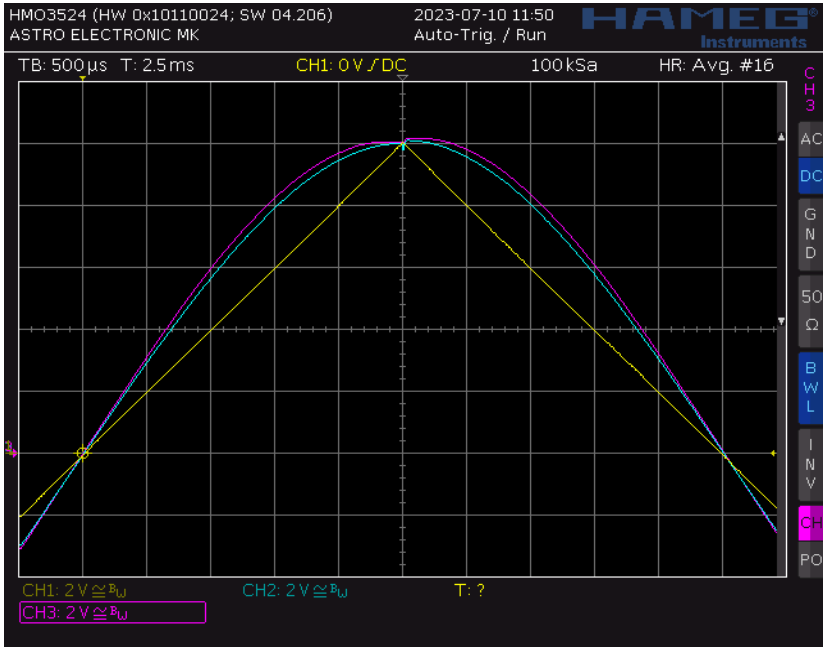




	3rd order sine approximation:	5th order sine approximation:
Equation and first derivative:	y = a · x + b · x^3 y' = a + 3b · x^2	y = a · x + b · x^3 + c · x^5 y' = a + 3b · x^2 + 5c · x^4
Condition 1: Peak amplitude = 1	y(1) = 1 a + b = 1	y(1) = 1 a + b + c = 1
Condition 2: Slope at peak = 0	y'(1) = 0 a + 3b = 0	y'(1) = 0 a + 3b + 5c = 0
Condition 3: Maximum slope = π/2	(only two conditions are possible)	y'(0) = π/2 a = π/2
Solution for coefficients:	a = 1.5 C2 = a/2 = 0.75 b = -0.5 C3 = -b = 0.5	a = π/2 = 1.571 b = 5/2-π = -0.642 c = (π-3)/2 = 0.071

Let's compare 3rd order (cyan) and 5th order (magenta) sine approximation. It's not a big difference:





The small step at the peak can be corrected by adding a very small amount of the square wave (from the comparator) to the approximated sine wave. Use a coefficient and a 100MΩ resistor.

See also: https://www.timstinchcombe.co.uk/index.php?pge=trisin

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 794

See also: Trigonometric Operations with the 433, in:

Analog Devices: The Best of Analog Dialogue, ISBN 0-9-916550-10-9, page 61 and 62

Using non-integer exponents:

sin(x) ≈ x - (x^2.827) / 6.28 (0 to π/2 radians) This has less than 0.25% error!

cos(x) ≈ 1 + 0.2325x - (x^1.504) / 1.445 (0 to π/2 radians) This has less than 1% error!

cos(x) = sin(π/2 - x) ≈ (π/2 - x) - ((π/2 - x)^2.827) / 6.28 (0 to π/2 radians) This has less than 0.25% error!

arctan(x) ≈ π/2 · x^1.2125 / (1 + x^1.2125)

Source: https://www.analog.com/media/en/analog-dialogue/volume-6/number-3/articles/volume6-number3.pdf#page=4

See also Bhāskara I's sine approximation formula: https://en.wikipedia.org/wiki/Bh%C4%81skara_I%27s_sine_approximation_formula

See also: https://www.analog.com/media/en/analog-dialogue/volume-18/number-1/articles/volume18-number1.pdf **Page 14**

See also (Tschebyscheff Approximation): https://rclab.de/analogrechner/tschebyscheffapproximation

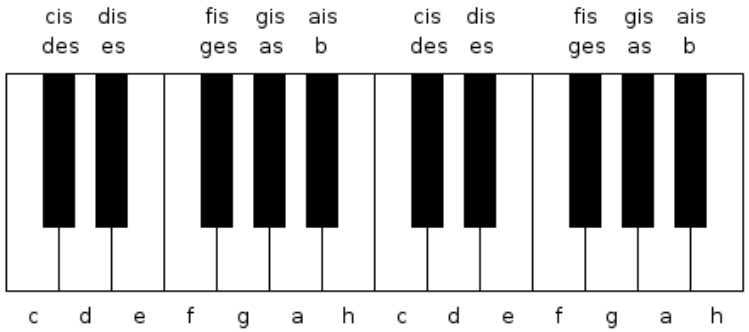
See also: Analog Devices: Nonlinear Circuits Handbook, Second Edition January 1976, page 60ff:

sin(x) ≈ 1.155 · x - 0.33 · x^2 +-2% error

sin(x) ≈ (1.0468 · x - 0.4278 · x^2) / (1 - 0.2618 · x) +-0.4% error

There are more approximations in the same chapter!

5.10 Keyboard with 1 V/Octave Signal, Triangle Wave Generator



If the integrator is used with an external input resistor connected to its SJ input, the resistor (in MΩ) can be calculated as follows:

R = 250 / f, where f is the frequency in Hz

The frequency can be increased by making the comparator limits smaller. Just insert a coefficient in the feedback from the comparator output to its input.

Frequency factor between two subsequent semitones: $2^{1/12} = 1.05946$

The resistors are calculated for a keyboard with 44 keys:

	c	cis/des	d	dis/es	e	f	fis/ges	g	gis/as	a	ais/b	h
Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
1						43.654 Hz 5.727 MΩ 5.257 MΩ	46.249 Hz 5.405 MΩ 4.935 MΩ	48.999 Hz 5.102 MΩ 4.632 MΩ	51.913 Hz 4.815 MΩ 4.345 MΩ	55.000 Hz 4.545 MΩ 4.075 MΩ	58.270 Hz 4.290 MΩ 3.820 MΩ	61.735 Hz 4.049 MΩ 3.579 MΩ
2	65.406 Hz 3.822 MΩ 3.352 MΩ	69.296 Hz 3.608 MΩ 3.138 MΩ	73.416 Hz 3.405 MΩ 2.935 MΩ	77.782 Hz 3.214 MΩ 2.744 MΩ	82.407 Hz 3.034 MΩ 2.564 MΩ	87.307 Hz 2.863 MΩ 2.393 MΩ	92.499 Hz 2.703 MΩ 2.233 MΩ	97.999 Hz 2.551 MΩ 2.081 MΩ	103.83 Hz 2.408 MΩ 1.938 MΩ	110.00 Hz 2.273 MΩ 1.803 MΩ	116.54 Hz 2.145 MΩ 1.675 MΩ	123.47 Hz 2.025 MΩ 1.555 MΩ
3	130.81 Hz 1.911 MΩ 1.441 MΩ	138.59 Hz 1.804 MΩ 1.334 MΩ	146.83 Hz 1.703 MΩ 1.233 MΩ	155.56 Hz 1.607 MΩ 1.137 MΩ	164.81 Hz 1.517 MΩ 1.047 MΩ	174.61 Hz 1.432 MΩ 961.6 kΩ	185.00 Hz 1.351 MΩ 881.3 kΩ	196.00 Hz 1.275 MΩ 805.4 kΩ	207.65 Hz 1.204 MΩ 733.9 kΩ	220.00 Hz 1.136 MΩ 666.3 kΩ	233.08 Hz 1.073 MΩ 602.5 kΩ	246.94 Hz 1.012 MΩ 542.3 kΩ
4	261.63 Hz 955.5 kΩ 485.5 kΩ	277.18 Hz 901.9 kΩ 431.9 kΩ	293.66 Hz 851.3 kΩ 381.3 kΩ	311.13 Hz 803.5 kΩ 333.5 kΩ	329.63 Hz 758.4 kΩ 288.4 kΩ	349.23 Hz 715.8 kΩ 245.8 kΩ	369.99 Hz 675.6 kΩ 205.6 kΩ	392.00 Hz 637.7 kΩ 167.7 kΩ	415.30 Hz 601.9 kΩ 131.9 kΩ	440.00 Hz 568.1 kΩ 98.15 kΩ	466.14 Hz 536.3 kΩ 66.26 kΩ	493.88 Hz 506.2 kΩ 36.16 kΩ
5	523.25 Hz 477.8 kΩ 7.75 kΩ											

First line: Frequency Second line: Resistance R = 250 / f, where f is the frequency in Hz

Third line: Resistance minus 470 kΩ, because it makes sense to place a fixed 470 kΩ series resistor close to the summing junction.

See also: "Making music" https://analogparadigm.com/downloads/alpaca_27.pdf

Finally I decided that it's better to work with a keyboard that makes 1 V/Octave signal. The keyboard simply contains a passive voltage divider which is connected to the keys. Of course you are only allowed to press one key at a time.

1 V/Octave, the voltage step from one to the next semitone is 1/12 V = 0.0833 V.

	c	cis/des	d	dis/es	e	f	fis/ges	g	gis/as	a	ais/b	h
Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
2						87.307 Hz 2.417 V	92.499 Hz 2.5 V	97.999 Hz 2.583 V	103.83 Hz 2.667 V	110.00 Hz 2.75 V	116.54 Hz 2.833 V	123.47 Hz 2.917 V
3	130.81 Hz 3.00 V	138.59 Hz 3.083 V	146.83 Hz 3.167 V	155.56 Hz 3.25 V	164.81 Hz 3.333 V	174.61 Hz 3.417 V	185.00 Hz 3.5 V	196.00 Hz 3.583 V	207.65 Hz 3.667 V	220.00 Hz 3.75 V	233.08 Hz 3.833 V	246.94 Hz 3.917 V
4	261.63 Hz 4.00 V	277.18 Hz 4.083 V	293.66 Hz 4.167 V	311.13 Hz 4.25 V	329.63 Hz 4.333 V	349.23 Hz 4.417 V	369.99 Hz 4.5 V	392.00 Hz 4.583 V	415.30 Hz 4.667 V	440.00 Hz 4.75 V	466.14 Hz 4.833 V	493.88 Hz 4.917 V
5	523.25 Hz 5.00 V	554.37 Hz 5.083 V	587.33 Hz 5.167 V	622.25 Hz 5.25 V	659.26 Hz 5.333 V	698.46 Hz 5.417 V	739.99 Hz 5.5 V	783.99 Hz 5.583 V	830.61 Hz 5.667 V	880.00 Hz 5.75 V	932.33 Hz 5.833 V	987.77 Hz 5.917 V
6	1046.5 Hz 6.00 V											

Voltage divider for 10 V:

1131 Ω + 43 · 39 Ω + 1872 Ω = 4680 Ω 2.136 mA at 10V 2.4167 V + 43 · 0.0833 V + 4.0000 V = 10.0000 V

$U = 4.75\ V + \log_2(f / 440\ Hz)$ $\log_2(x) = \ln(x) / \ln(2)$ --> $U = 4.75\ V + \ln(f / 440\ Hz) / \ln(2)$ $U = 4.75\ V + 1.4427 \cdot \ln(f / 440\ Hz)$

$f = 440\ Hz \cdot 2^{(U - 4.75\ V)} = 16.3516\ Hz \cdot 2^U$ In these formulas U is the control voltage in V

Frequency of triangle wave generator, where the "10" input of the integrator is used:

f = C · 2500 Hz, where C is the coefficient at the input of the integrator, or the control voltage at the inmput of a multiplier.

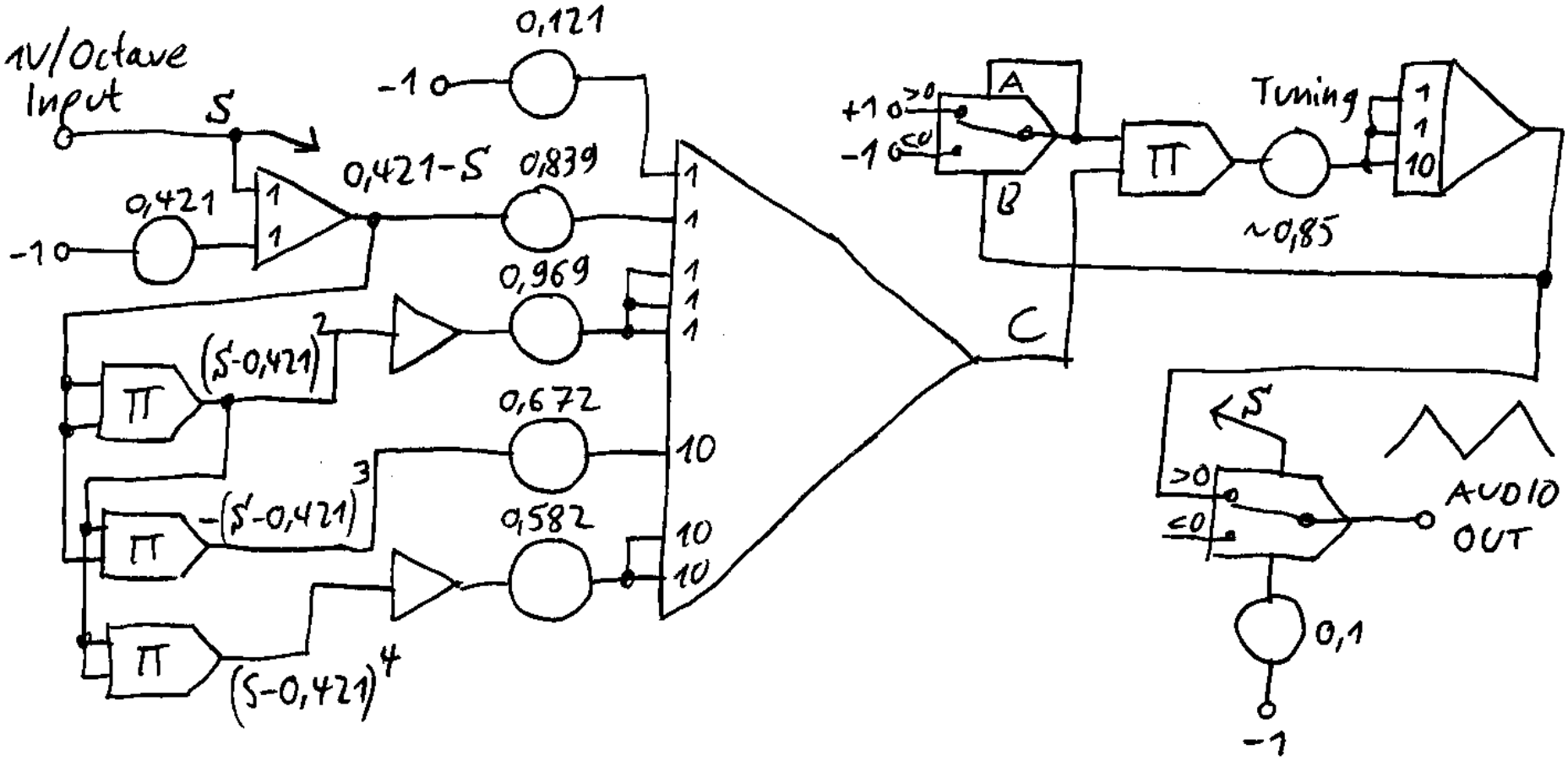
Calculate C as a function of the 1 V/octave control signal in machine units: S = U / 10

$C = 16.3516 \cdot 2^{10S} / 2500 = 0.006541 \cdot 2^{10S}$ S = 0.2417 ... 0.421 ... 0.6000 (0.421 is in the middle of the range)

Taylor series (from Wolfram Alpha): C = 0.121054 + 0.839085 (S - 0.421) + 2.90805 (S - 0.421)^2 + 6.71901 (S - 0.421)^3 + 11.6432 (S - 0.421)^4

Use this text as input for Wolfram Alpha: "series of y=0.006541*2^(10*x) at x=0.421"

This circuit gets a 1 V/Octave input signal from a keyboard and makes a triangular wave. This is version 1: **Not good, don't use it !**



My measurement results (the oscillator was calibrated at 440Hz):

Note	Frequency Hz	Control voltage S		C (calculated) in machine units				Frequency (calculated) (Error > 1%) (Error > 2%)			Frequency (measured) (Error > 1%) (Error > 2%)			
		Volt	Machine units	2nd order	3rd order	4th order	exact	2nd order Hz	3rd order Hz	4th order Hz	Analog 4th order, Version 1 Hz	Analog 4th order, Version 2 Hz	Analog 4th order, Version 3 Hz	Digital 12-bit look-up-table in Teensy LC Hz
F2	87.307	2.417	0.2417	0.0641	0.0254	0.0374	0.0349	160.24	63.414	93.498	106	96.2	92.9	86.8
A2	110.00	2.75	0.275	0.0605	0.0396	0.0449	0.0440	151.34	99.063	112.29	124	114.8	112.0	109.7
C3	130.81	3.00	0.300	0.0621	0.0502	0.0527	0.0523	155.25	125.50	131.74	143	134.1	131.9	130.7
F3	174.61	3.417	0.3417	0.0728	0.0695	0.0699	0.0698	182.00	173.63	174.78	184	177.2	176.0	175.2
A3	220.00	3.75	0.375	0.0886	0.0880	0.0880	0.0880	221.52	219.89	220.02	228	222.4	221.9	220.9
C4	261.63	4.00	0.400	0.1047	0.1047	0.1047	0.1047	261.79	261.63	261.64	269	263.5	263.5	262.4
F4	349.23	4.417	0.4417	0.1397	0.1397	0.1397	0.1397	349.17	349.32	349.33	354	350.0	350.2	349.7
A4	440.00	4.75	0.475	0.1748	0.1759	0.1760	0.1760	437.11	439.76	440.00	440	440.0	440.0	440.0
C5	523.25	5.00	0.500	0.2055	0.2088	0.2093	0.2093	513.73	522.01	523.14	519	520.6	520.4	522.9
F5	698.46	5.417	0.5417	0.2647	0.2765	0.2790	0.2794	661.74	691.28	697.46	683	688.8	688.2	694.4
A5	880.00	5.75	0.575	0.3192	0.3438	0.3503	0.3520	798.10	859.45	875.82	850	859.1	858.5	872.2
C6	1046.5	6.00	0.600	0.3644	0.4030	0.4149	0.4186	911.07	1007.41	1037.29	1000	1010	1010	1032

Although the errors of the Taylor series are already quite large, most of the errors in version 1 came from inaccuracies in the analog computation. Let's analyze what's the problem. The term (S - 0.421) is always in the range from -0.1793 to +0.179. If this variable is raised to a power, it becomes very small. It becomes so small, that it's domonated by the offset errors of the analog multipliers. And then the variable (and the error) is multiplied by quite large coefficients, the largest one is 11.643. This Taylor series is mathematically correct, but it's not optimized for the analog computer!

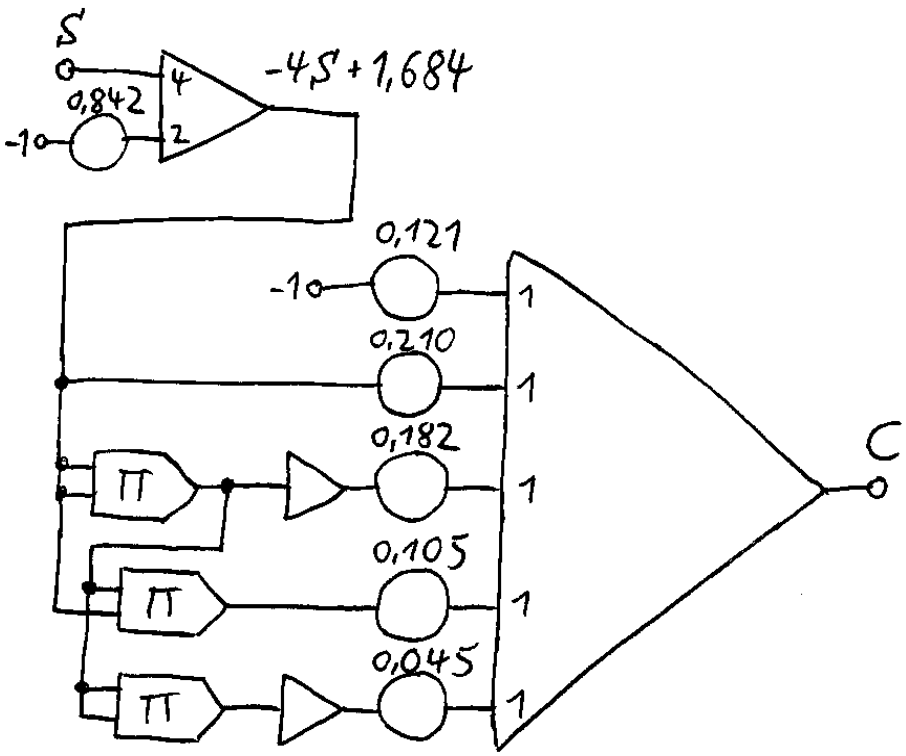
C = 0.121 + 0.839 (S - 0.421) + 2.908 (S - 0.421)^2 + 6.719 (S - 0.421)^3 + 11.643 (S - 0.421)^4 Version 1, not optimized for analog computer !

How can the Taylor series be optimized?

Let's multiply the term (S - 0.421) by 4 (even 5 would have been possible). Then it's always in the range from -0.7172 to +0.716. When this variable is raised to the 4th power, it's still 0.26. To compensate for this change, we must divide the coefficient in front of the term by 4 raised to the corresponding power. Now we have this Taylor series, which is mathematically exactly the same as before, but it's much less sensitive to offset errors in the multipliers:

C = 0.121 + 0.210 · (4S - 1.684) + 0.182 (4S - 1.684)^2 + 0.105 (4S - 1.684)^3 + 0.045 (4S - 1.684)^4 Version 2, optimized for analog computer !

This is the circuit for the improved version 2:



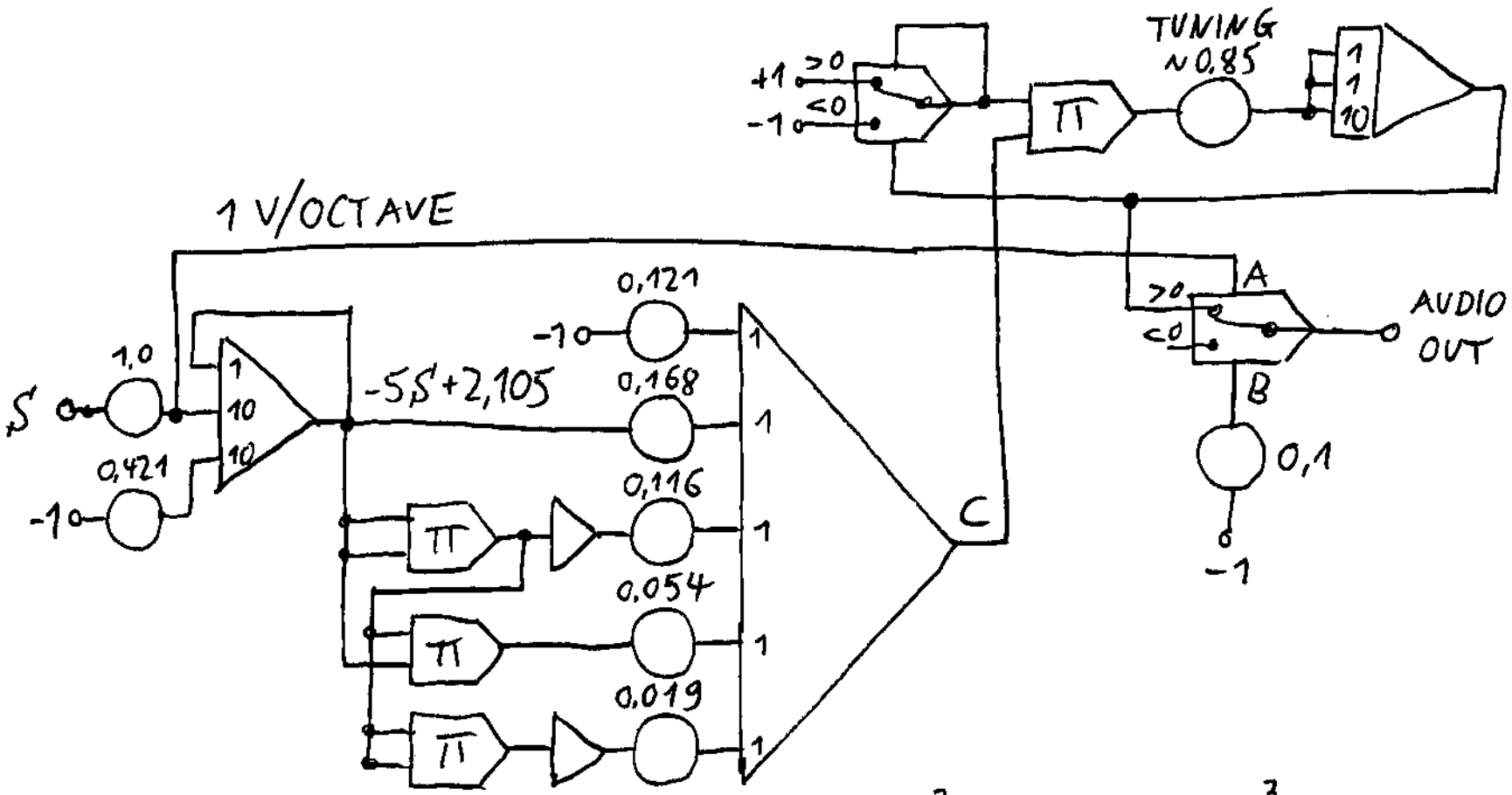
Can it be further improved? Yes, there are two things that can be improved:

First, the term (S - 0.421) can be multiplied by 5 instead of 4. The result still fits in the -1 to +1 machine unit range.

Second, the input signal comes from a passive resistor divider in the keyboard, which is not a low impedance source. It should go to a high impedance input. In version 2 it did go to 5 inputs with 1MΩ input resistance each, so the total input impedance was 200kΩ. This can be improved by inserting a coefficient which is set to 1.000. Then the input impedance is 1MΩ.

C = 0.121 + 0.168 · (5S - 2.105) + 0.116 (5S - 2.105)^2 + 0.054 (5S - 2.105)^3 + 0.019 (5S - 2.105)^4 **Version 3, optimized for analog computer !**

This is version 3:



$$C = 0{,}121 + 0{,}168(5{,}5'-2{,}105) + 0{,}116(5{,}5'-2{,}105)^2 + 0{,}054(5{,}5'-2{,}105)^3 + 0{,}019(5{,}5'-2{,}105)^4$$

For the last column the function C = f(S) was realized as a digital look-up-table in Teensy LC, with 12-bit input and output. A 19th order Taylor series was used as an approximation for the exponential function, because the built-in exp() function needs too much memory:

```
float function(float x)   // This is the user-defined function with input and output in machine units
{
  return(0.006541 * expo(6.931471806 * x);
}
```

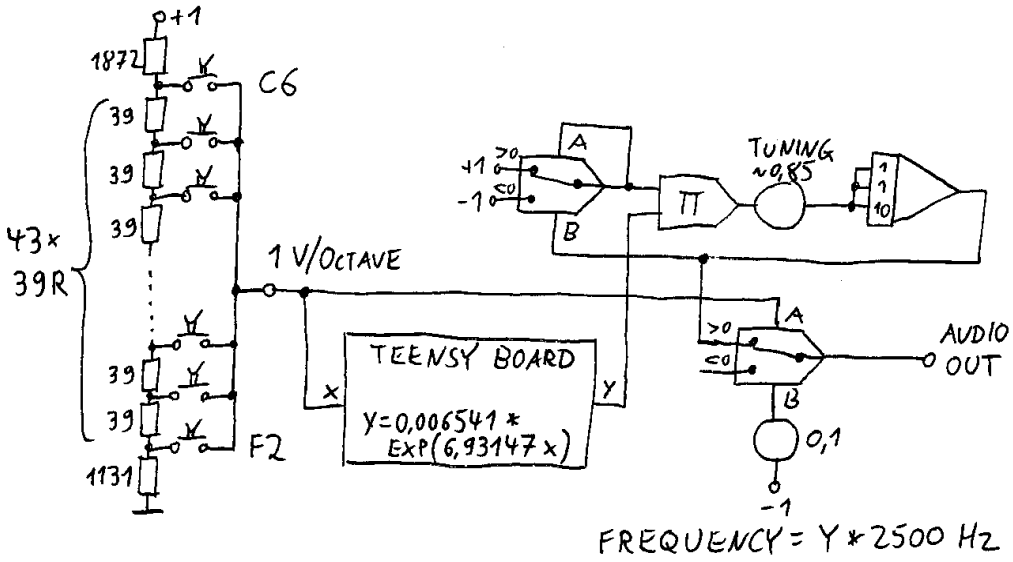
```
float expo(float x)   // This is a power series approximation for the exponential function,
{                     // because the built-in exponential function needs too much RAM
  float y = 1;
  float z = 1;
  float n = 1;
  for (int i = 1; i < 20; i++)
  {
    z = z * x;
    n = n * i;
    y = y + z / n;
  }
  return(y);
}
```

$C = 0.006541 \cdot 2^{10S}$

$= 0.006541 \cdot \exp(10 \cdot x \cdot \ln(2))$

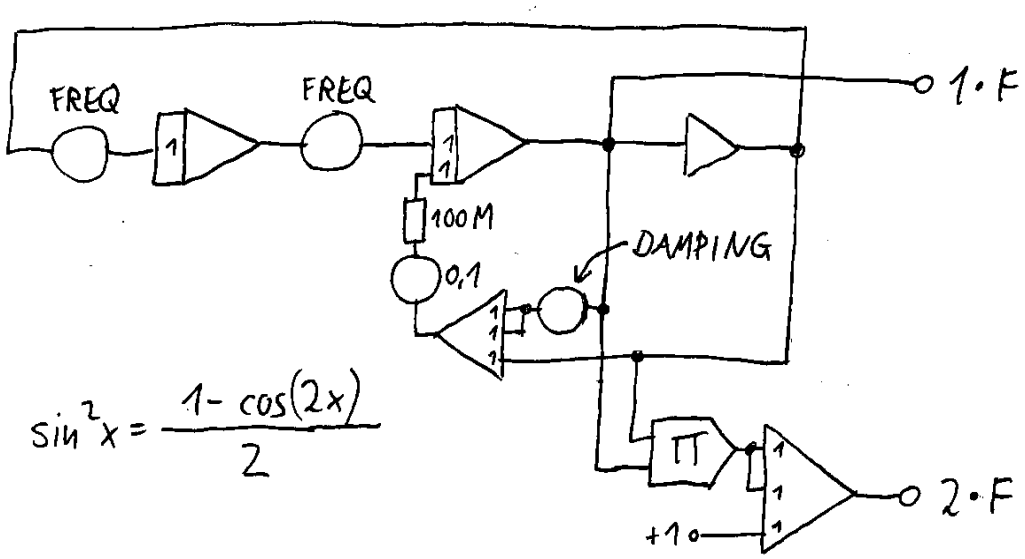
$= 0.006541 \cdot \exp(6.931471806 \cdot x)$

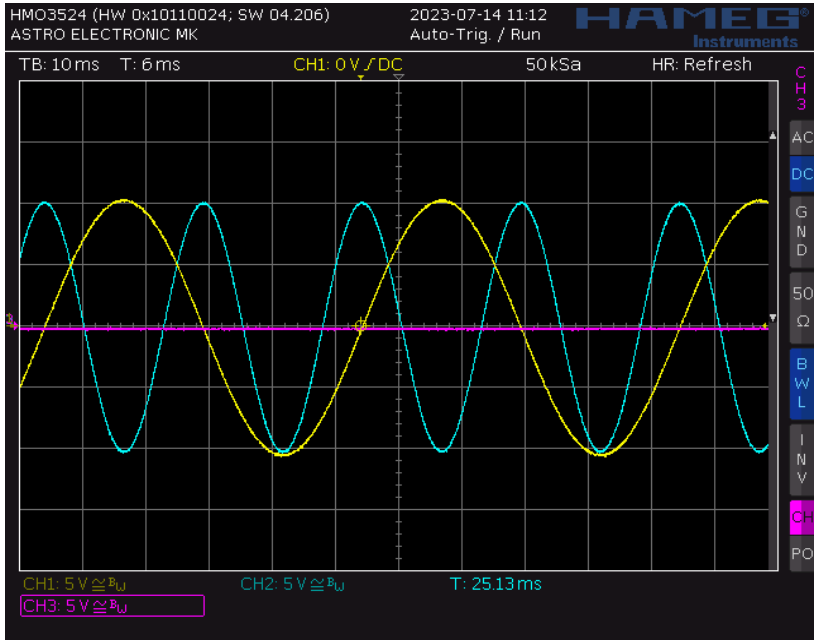
This is the circuit with the Teensy LC board as a look-up-table:



5.11 Frequency Doubling for Sine Waves

Forgot the initial condition for one integrator in the schematic.



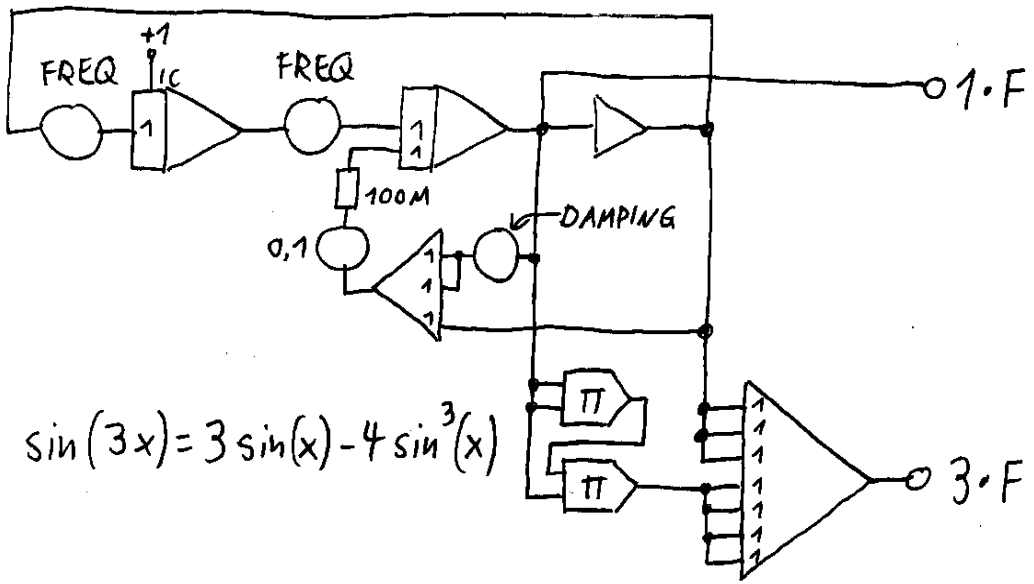


It's possible to use one of these formulas:

$\cos(2x) = 1 - 2 \cdot \sin^2(x)$

$\sin(2x) = 2 \cdot \sin(x) \cdot \cos(x)$

5.12 Frequency Tripling for Sine Waves



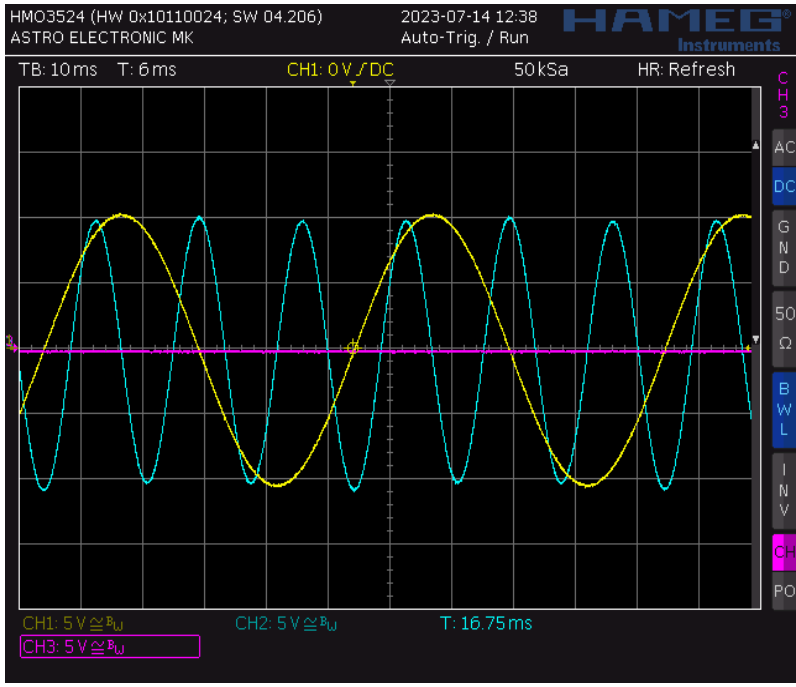
It's possible to use one of these formulas:

$\sin(3x) = \sin(x) \cdot (1 + 2\cos(2x))$

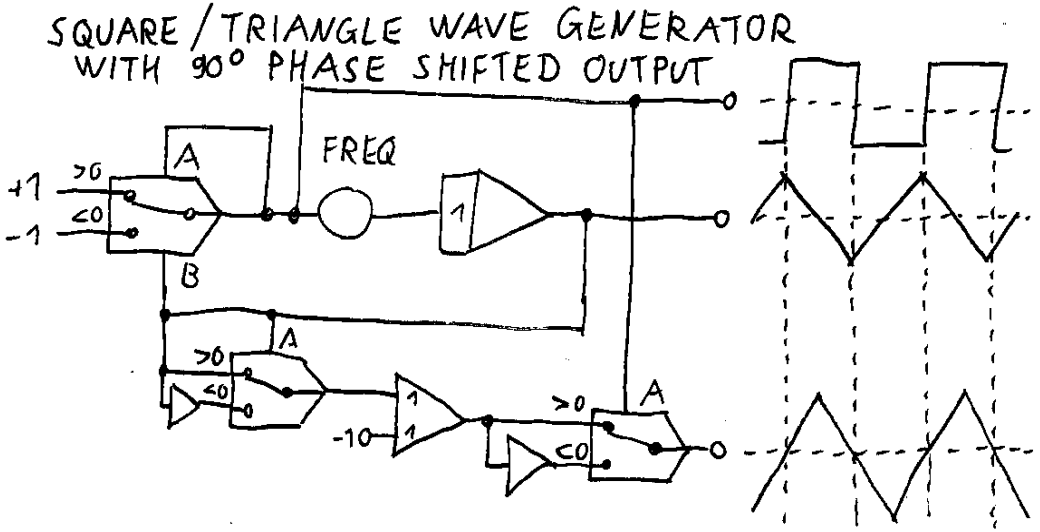
$\sin(3x) = 3 \cdot \sin(x) - 4 \cdot \sin^3(x)$

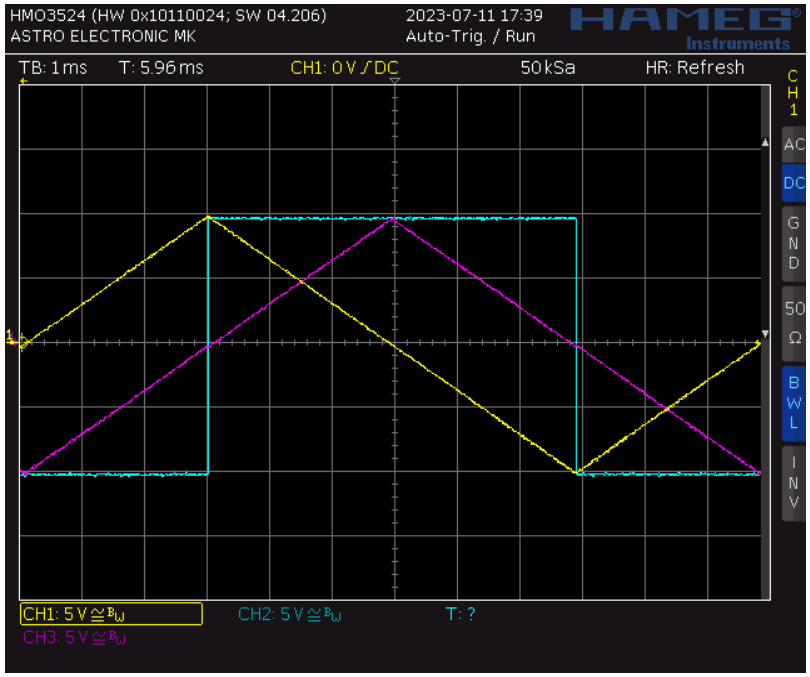
$\cos(3x) = \cos(x) \cdot (2 \cdot \cos(2x) - 1)$

$\cos(3x) = 4 \cdot \cos^3(x) - 3 \cdot \cos(x)$



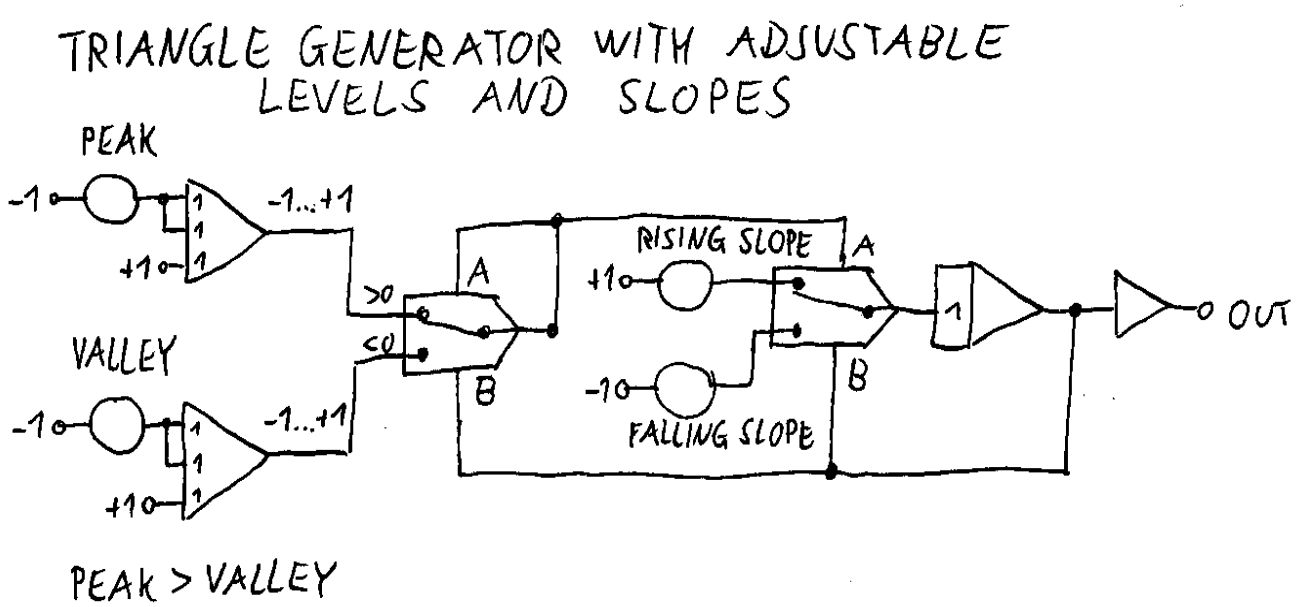
5.13 Square and Triangle Wave Generator with 90° Phase Shift



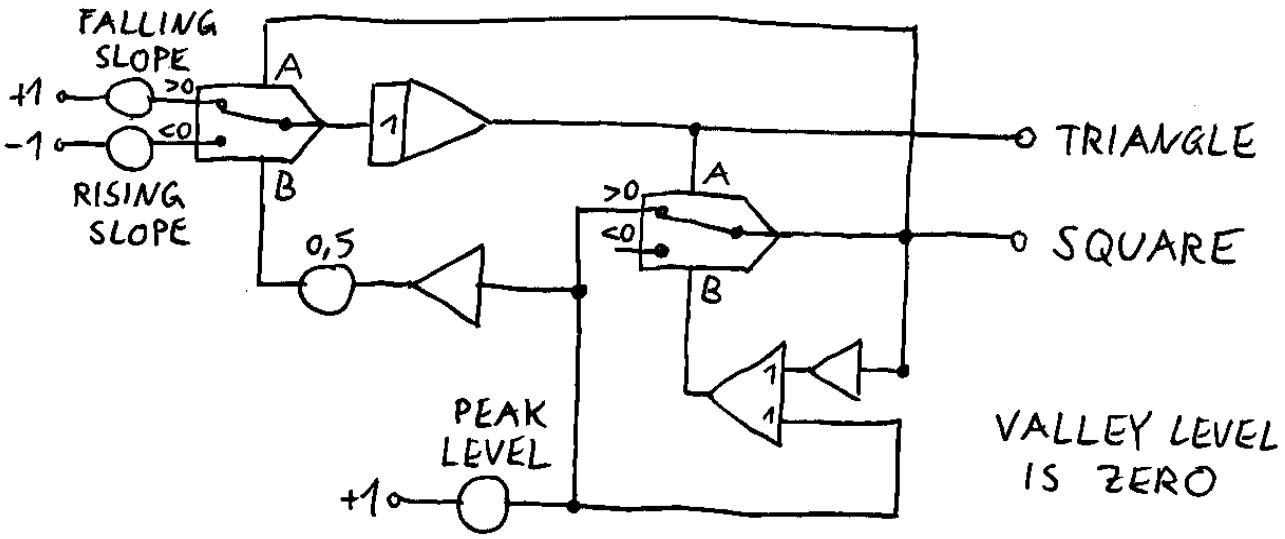


5.14 Triangle Generator with adjustable Levels and Slopes

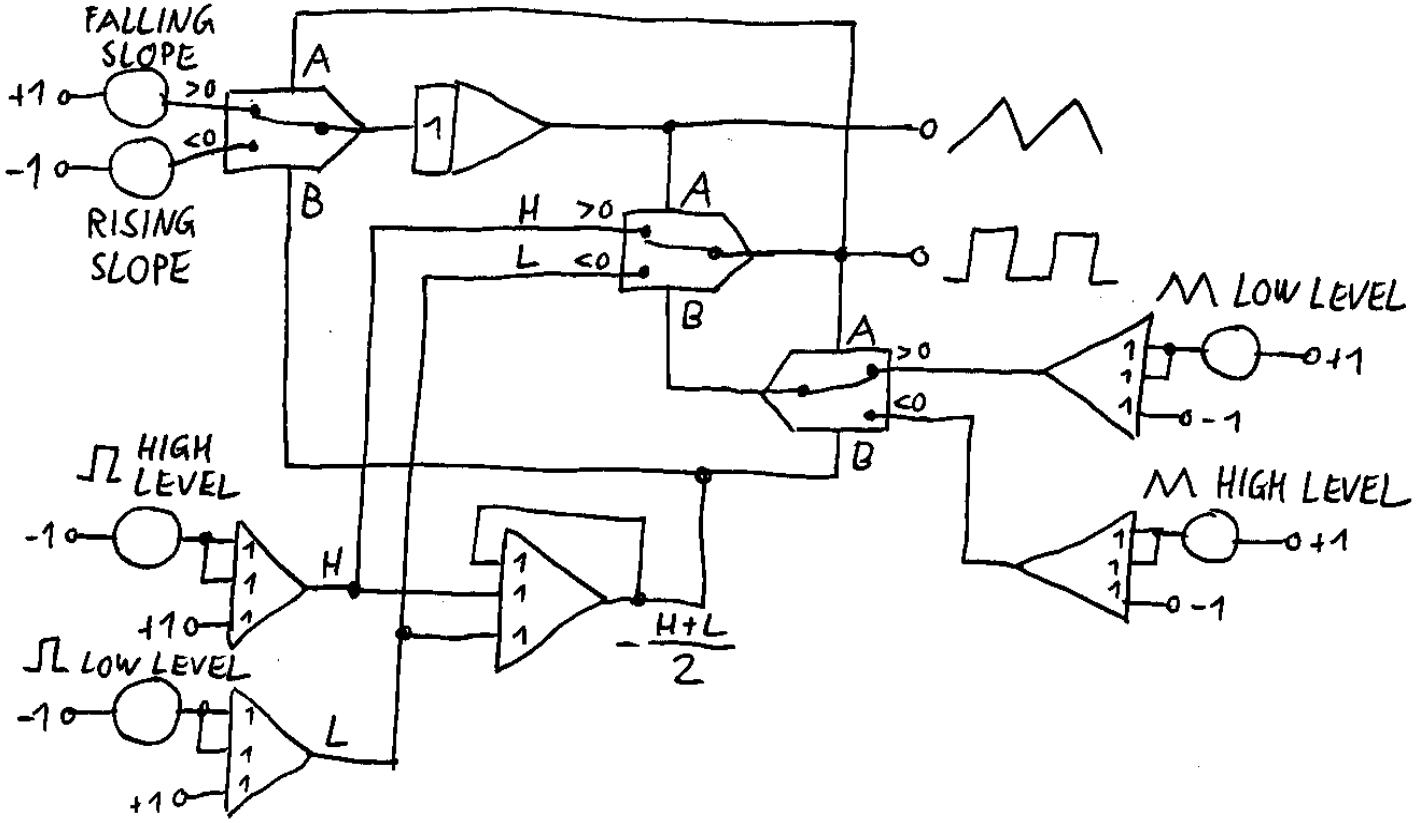
Four parameters can be adjusted: Peak level, valley level, rising slope and falling slope. Peak level must not be smaller than valley level. Peak and valley can also be both positive or both negative.



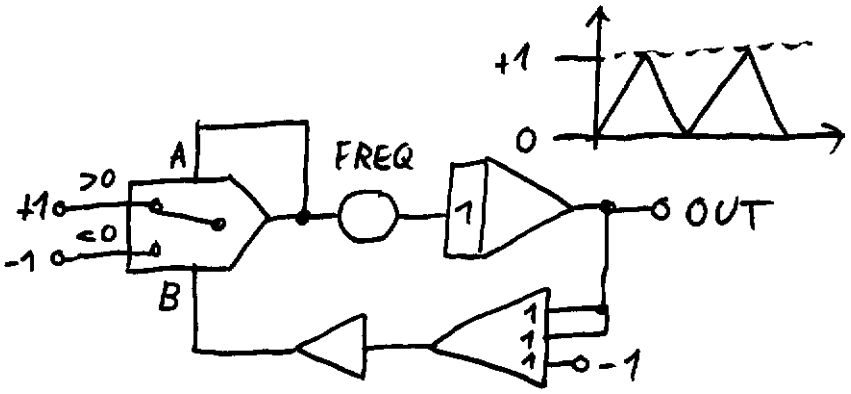
This triangle / square wave generator has an adjustable peak level (for both waveforms) and the valley level is zero (for both waveforms):



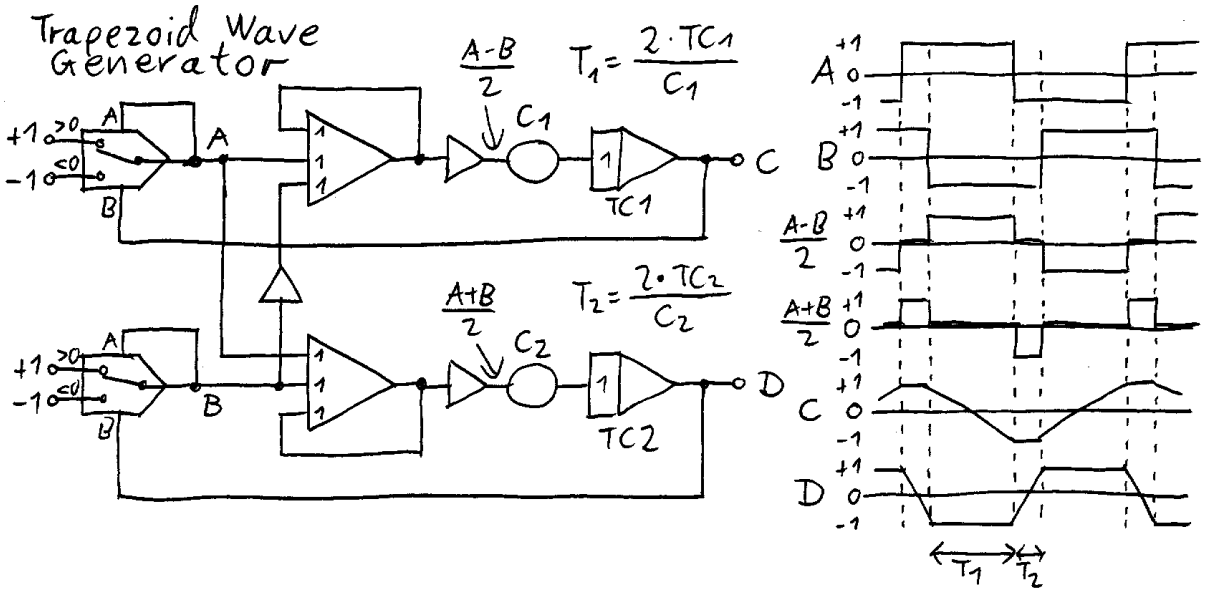
In this triangle / square wave generator everything is adjustable:

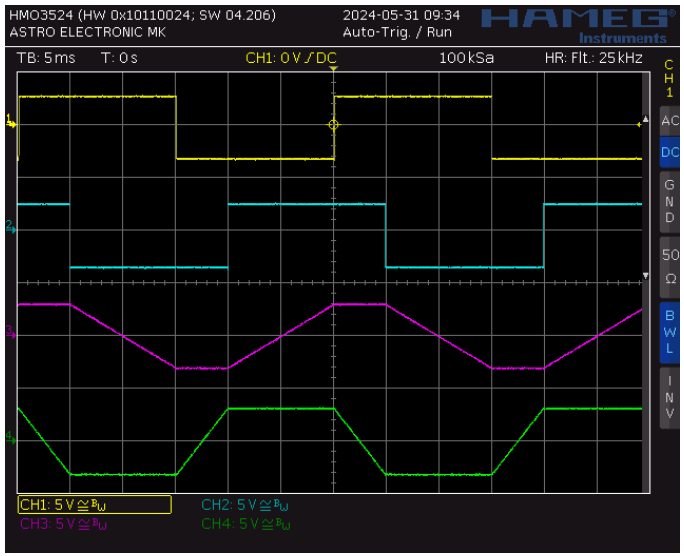


In this triangle / square wave generator the triangle is between 0 and 1:

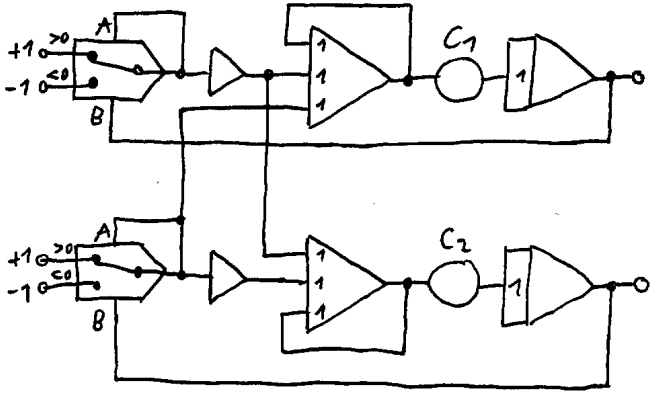


5.15 Trapezoid Wave Generator

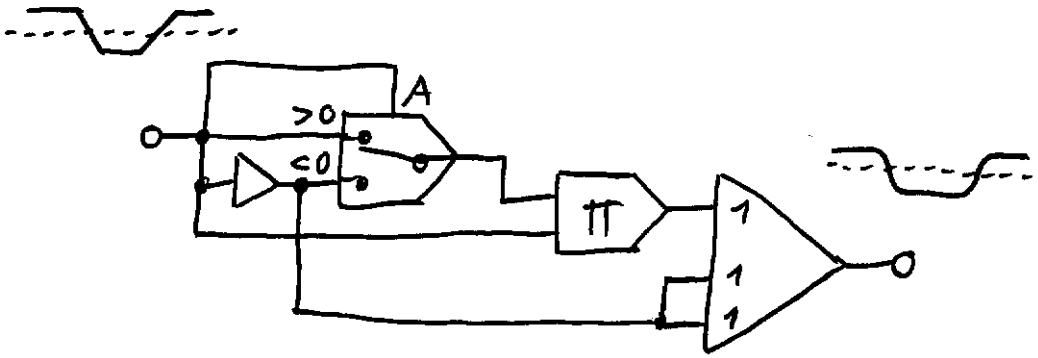


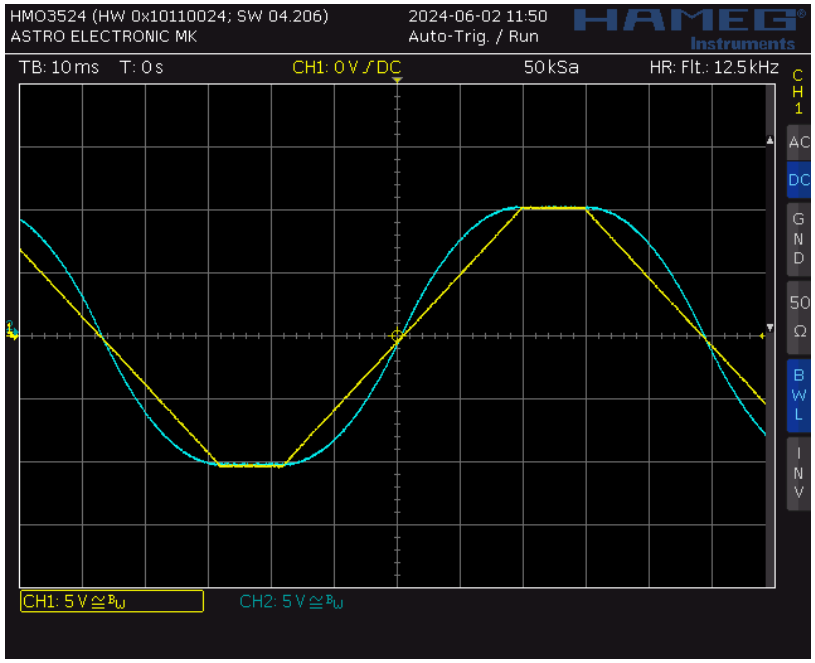


It can be simplified a little bit, needing one inverter less:



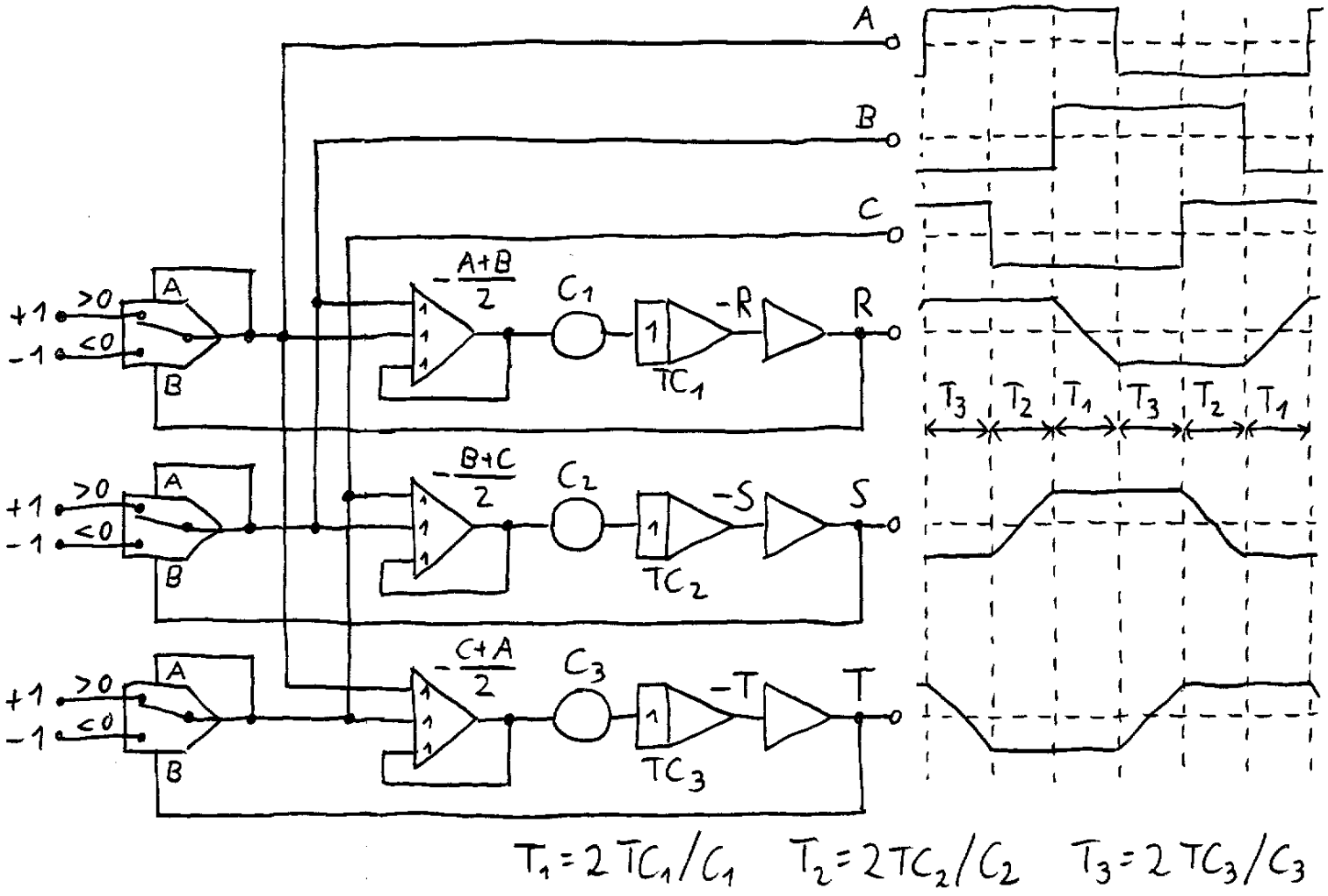
The linear ramps can be replaced by S-curves with this circuit:



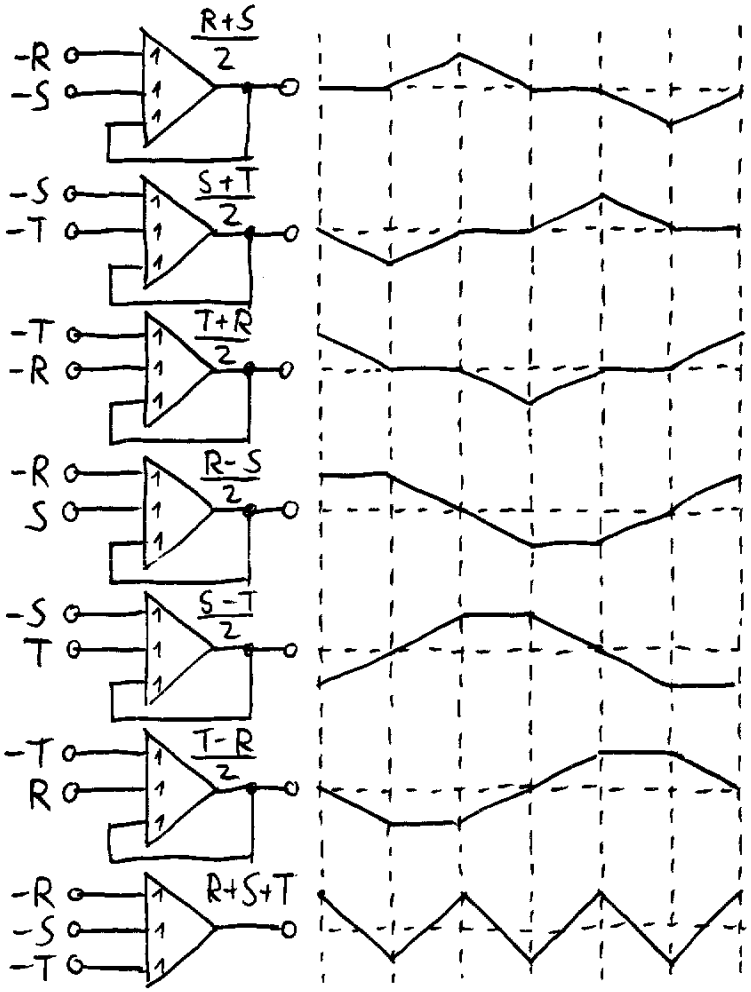


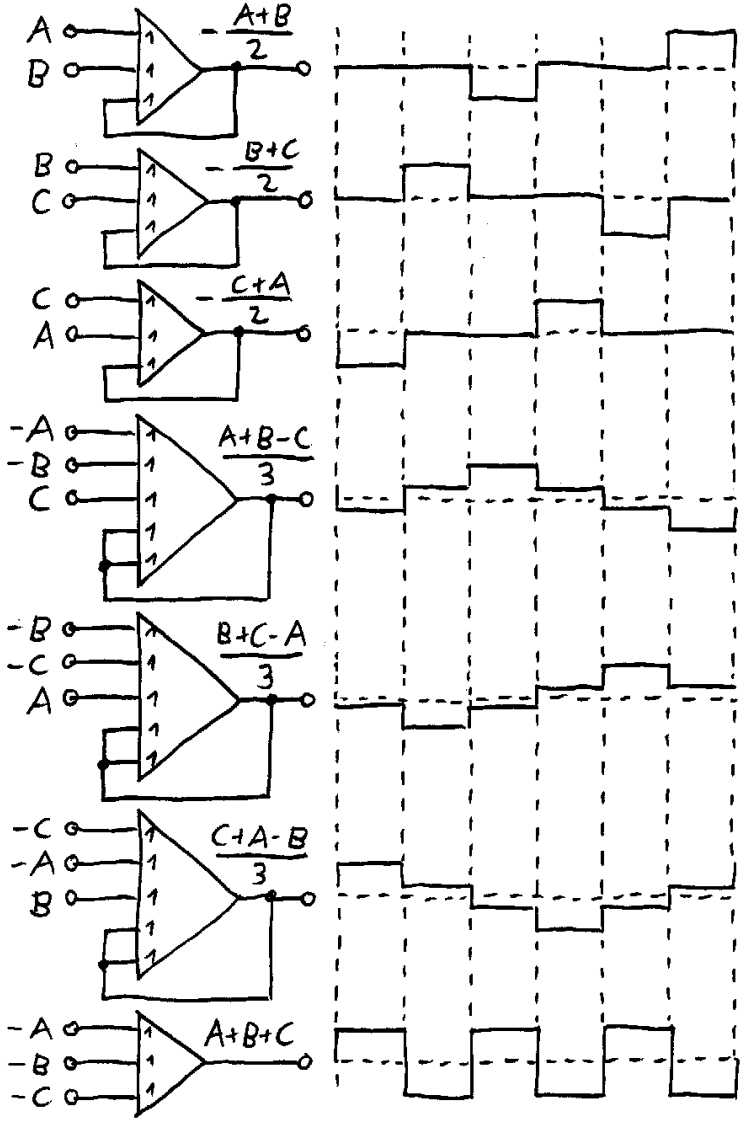
5.16 Trapezoid Wave Generator, 3-Phase Version

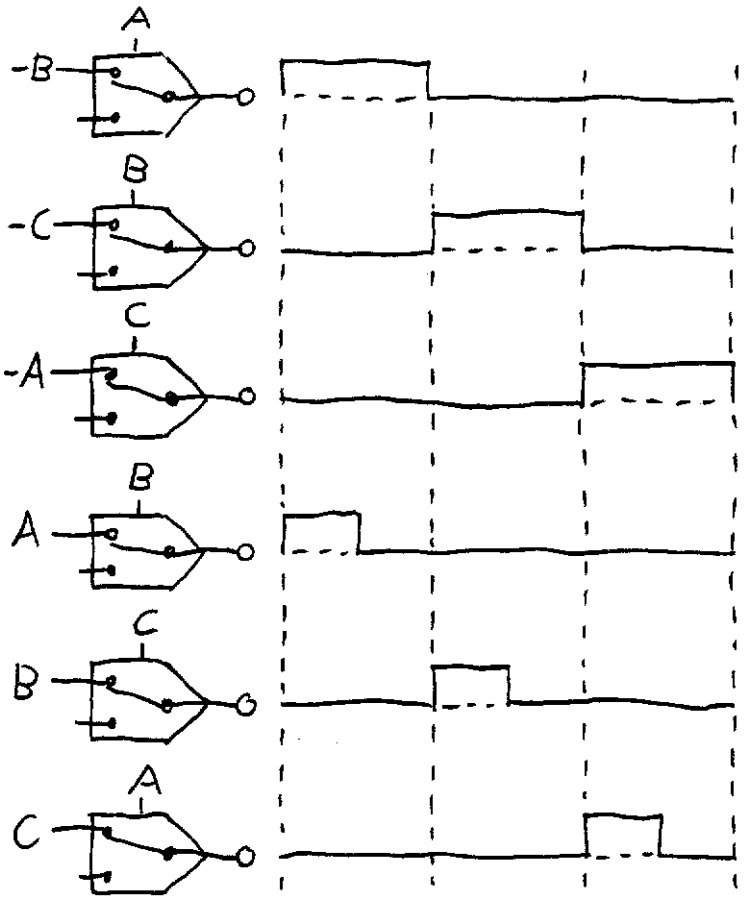
It's also possible to make a 3-phase trapezoid wave generator.
The durations T_1, T_2 and T_3 can be set indepentently of each other.

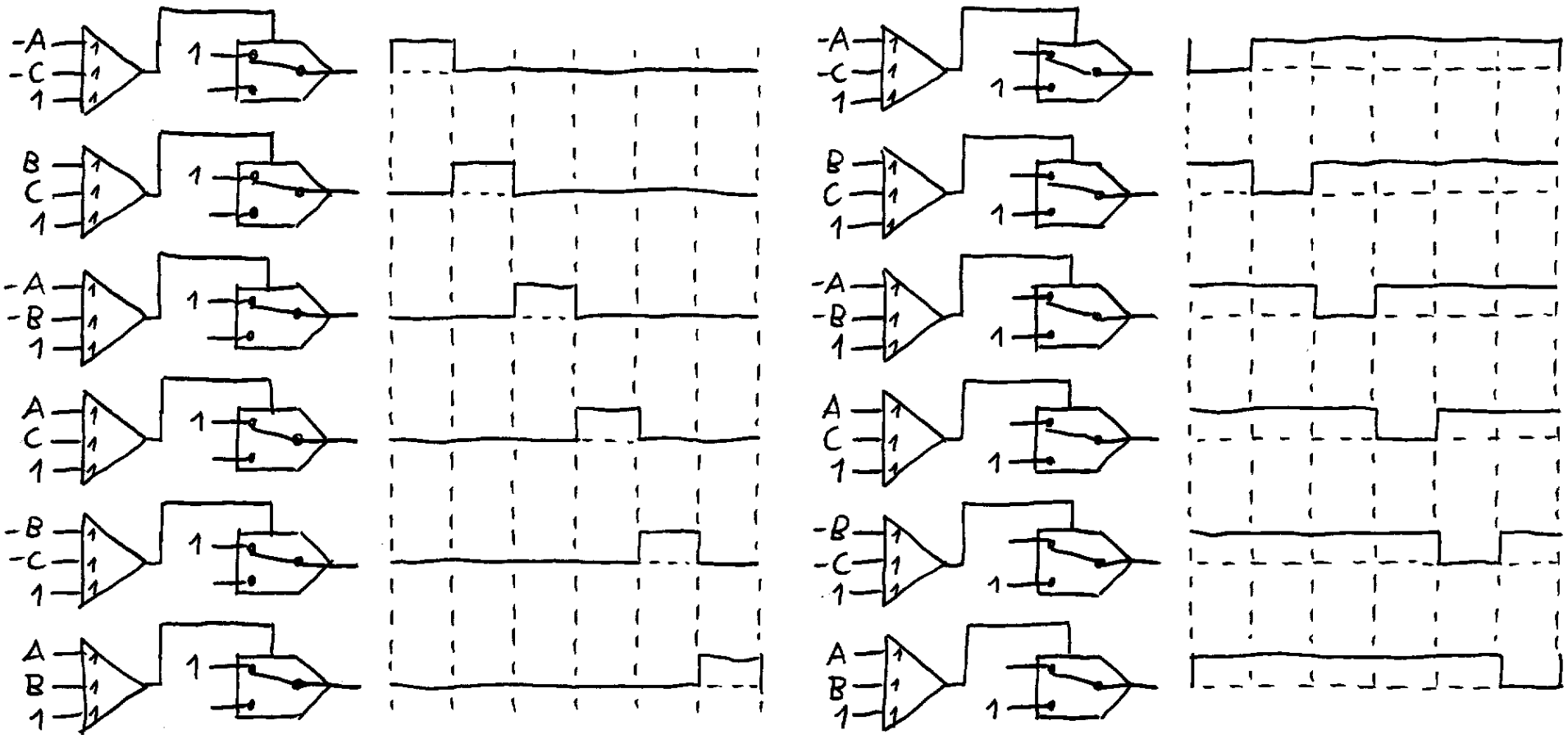


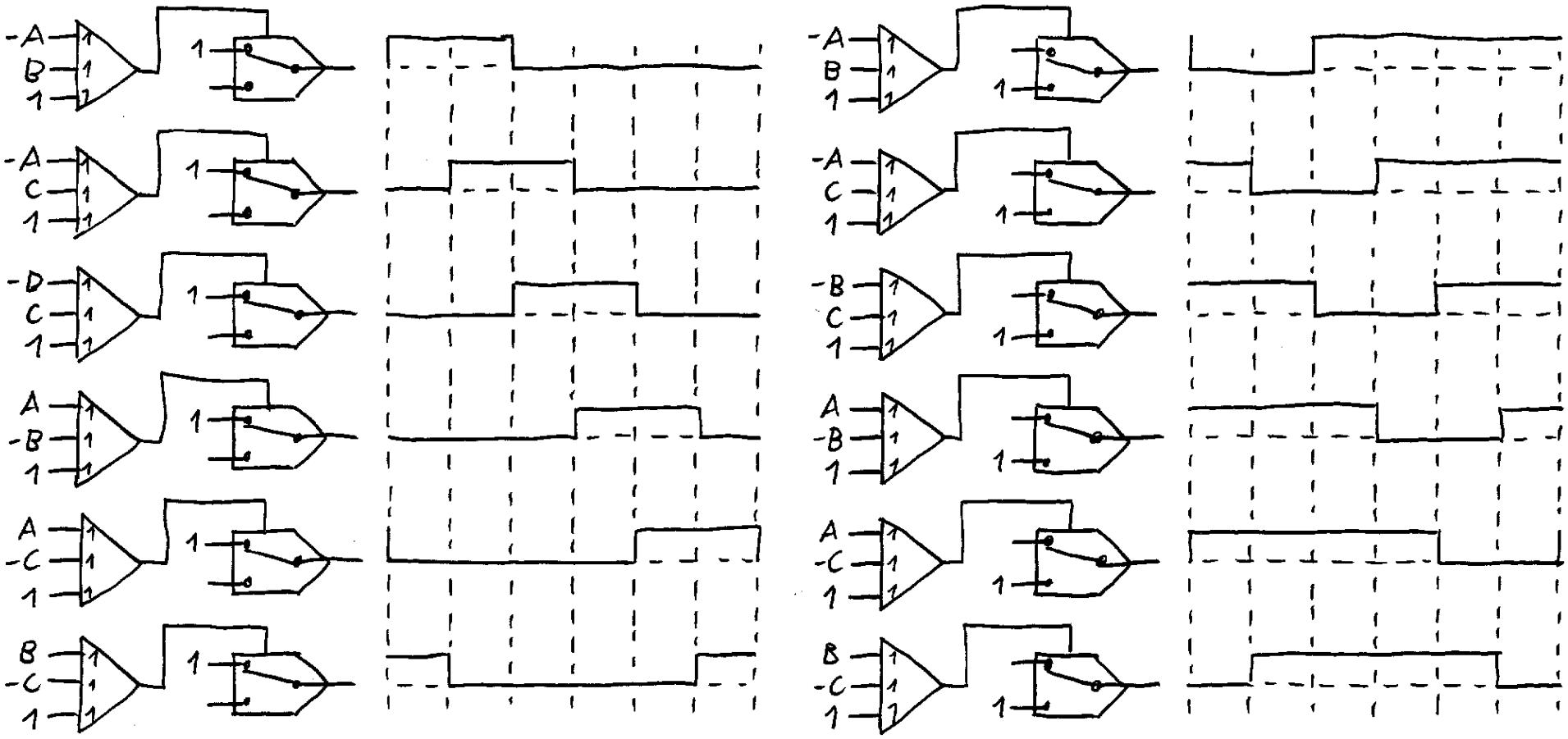
Other waveforms can be derived from the R, S, T and A, B, C signals:



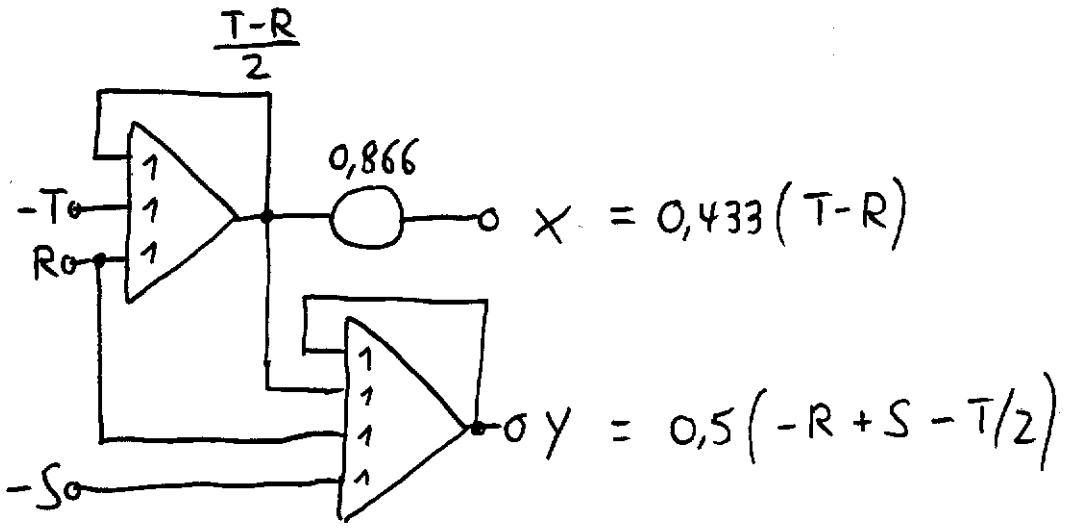



A
-B
B
-C
C
-A
B
A
C
B
A
C


-A
-C
1
1
B
C
1
-A
-B
1
A
C
1
-B
-C
1
A
B
1

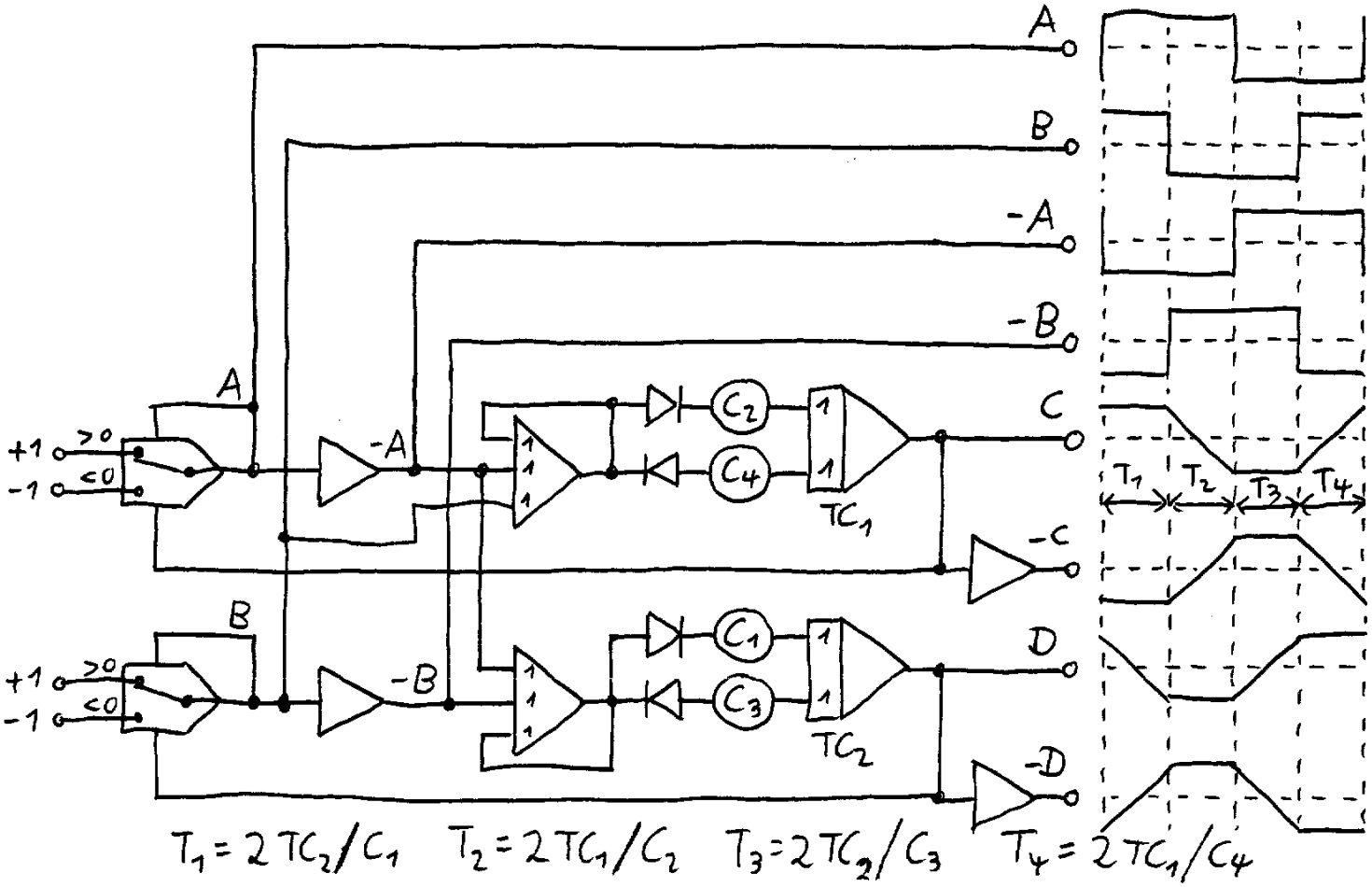

-A
B
1
-A
C
1
-D
C
1
A
-B
1
A
-C
1
B
-C
1
-A
B
1
-A
C
1
-B
C
1
A
-B
1
A
-C
1
B
-C
1

This is a generator for a hexagon in the x,y plane:

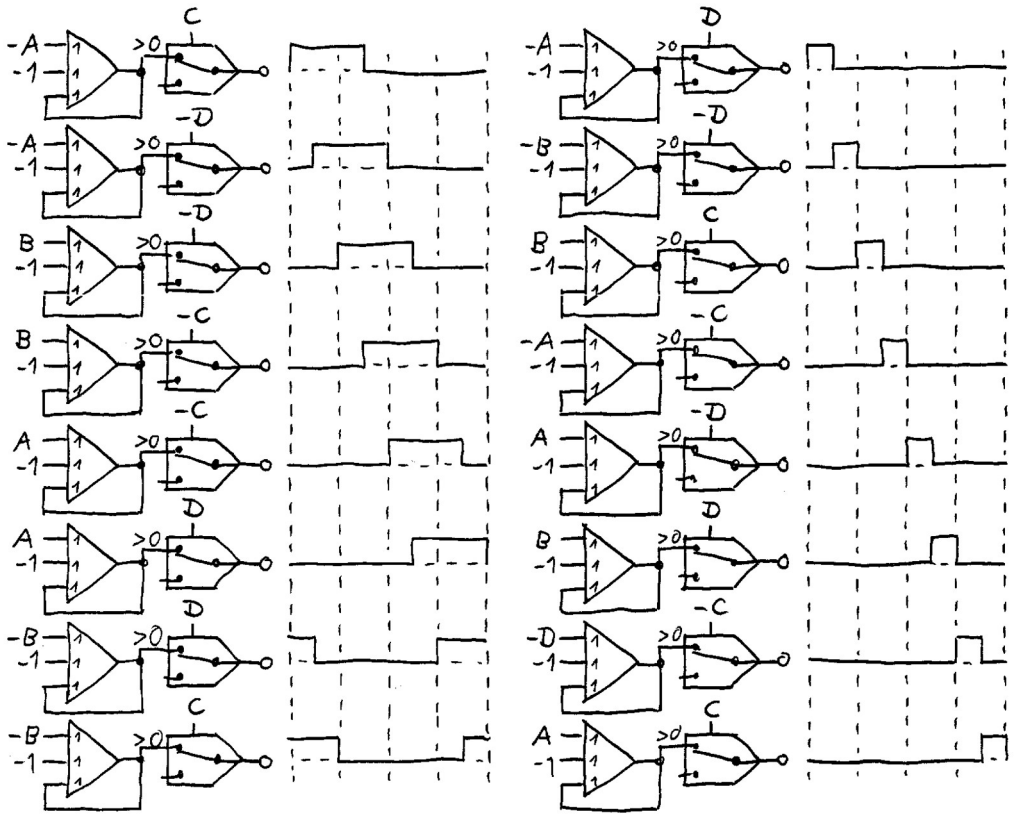


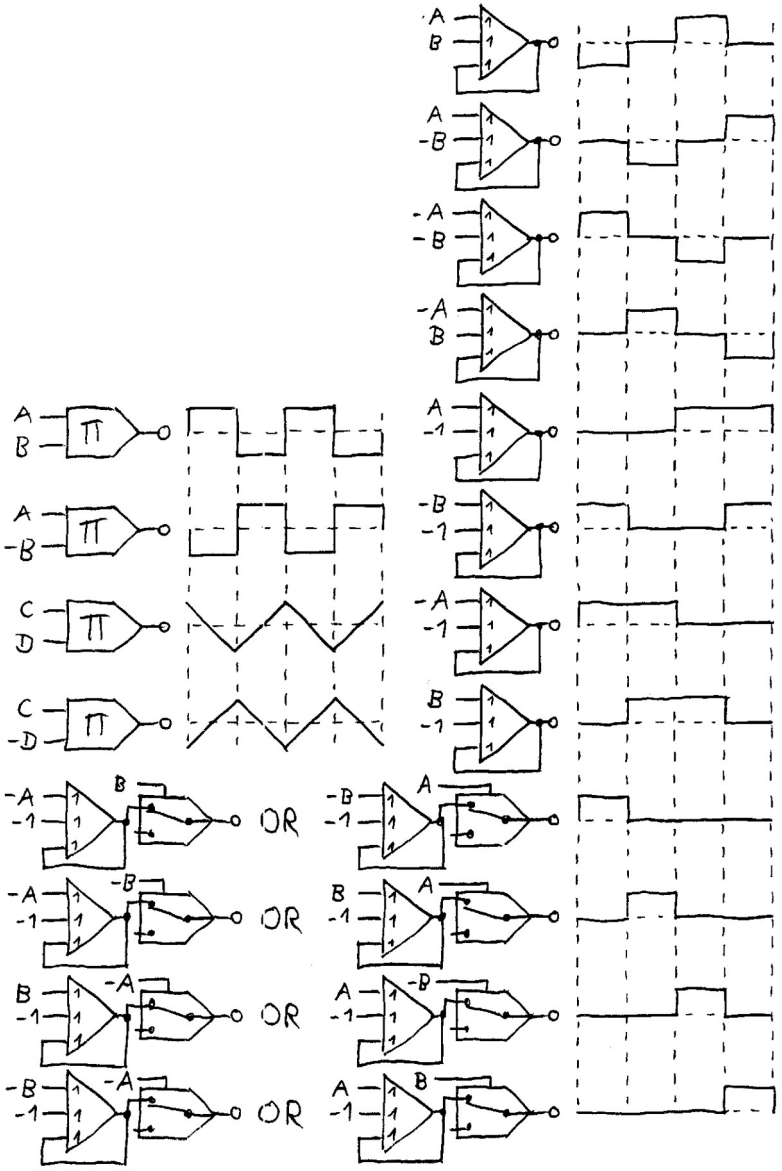
5.17 4-Step Sequencer

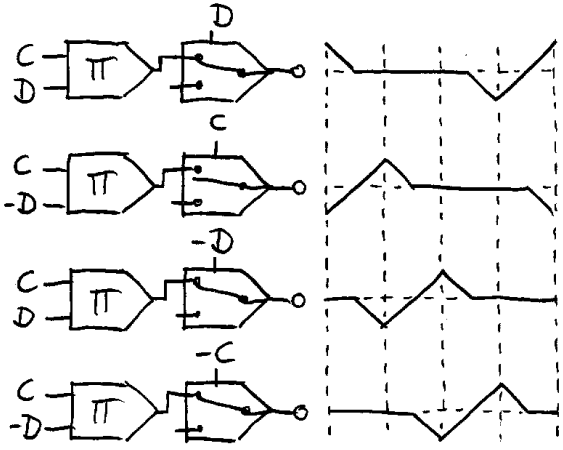
In this 4-step sequencer all 4 durations T1, T2, T3 and T4 can be set independently of each other. This is a toolbox for making many different waveforms:

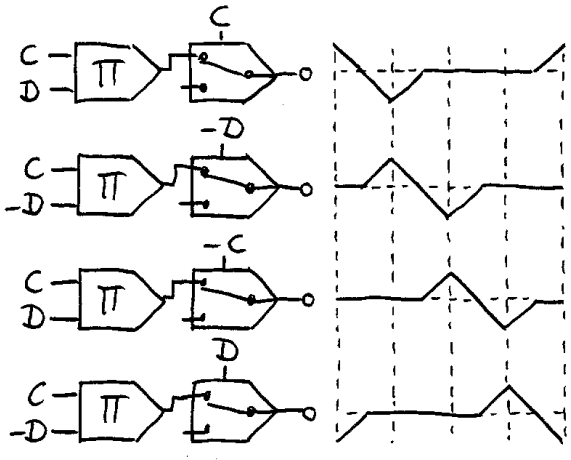


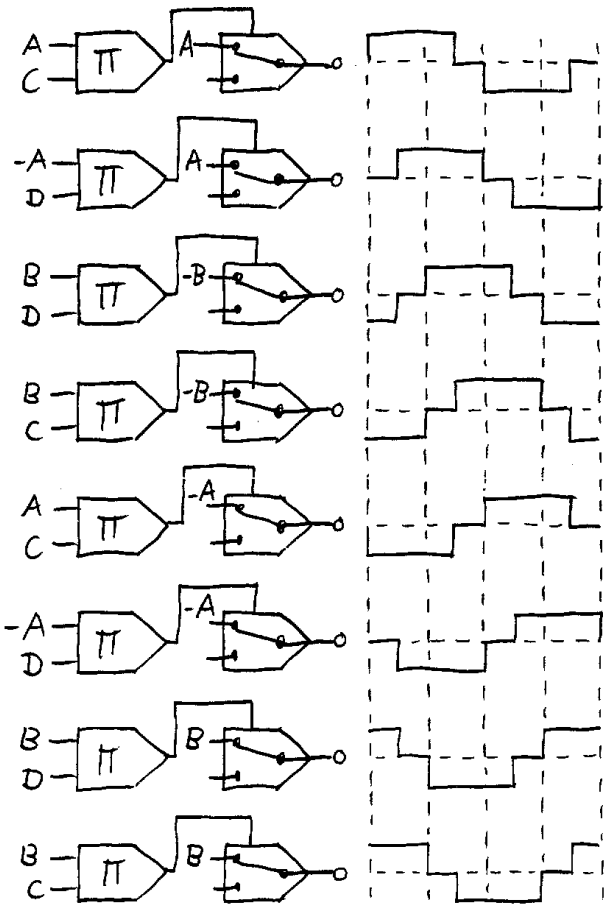
These (and many more) waveforms can be derived from the A, B, C and D signals:

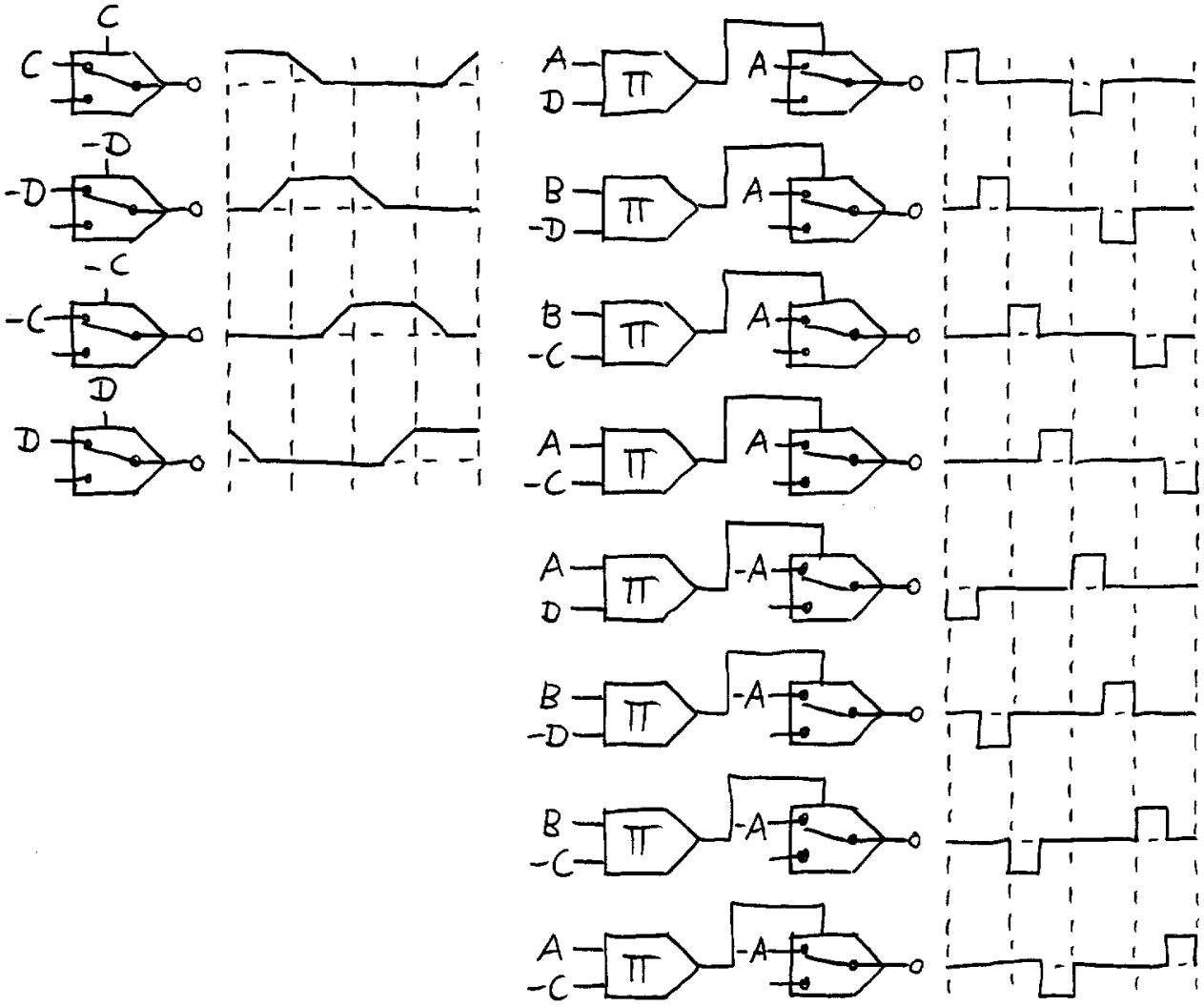


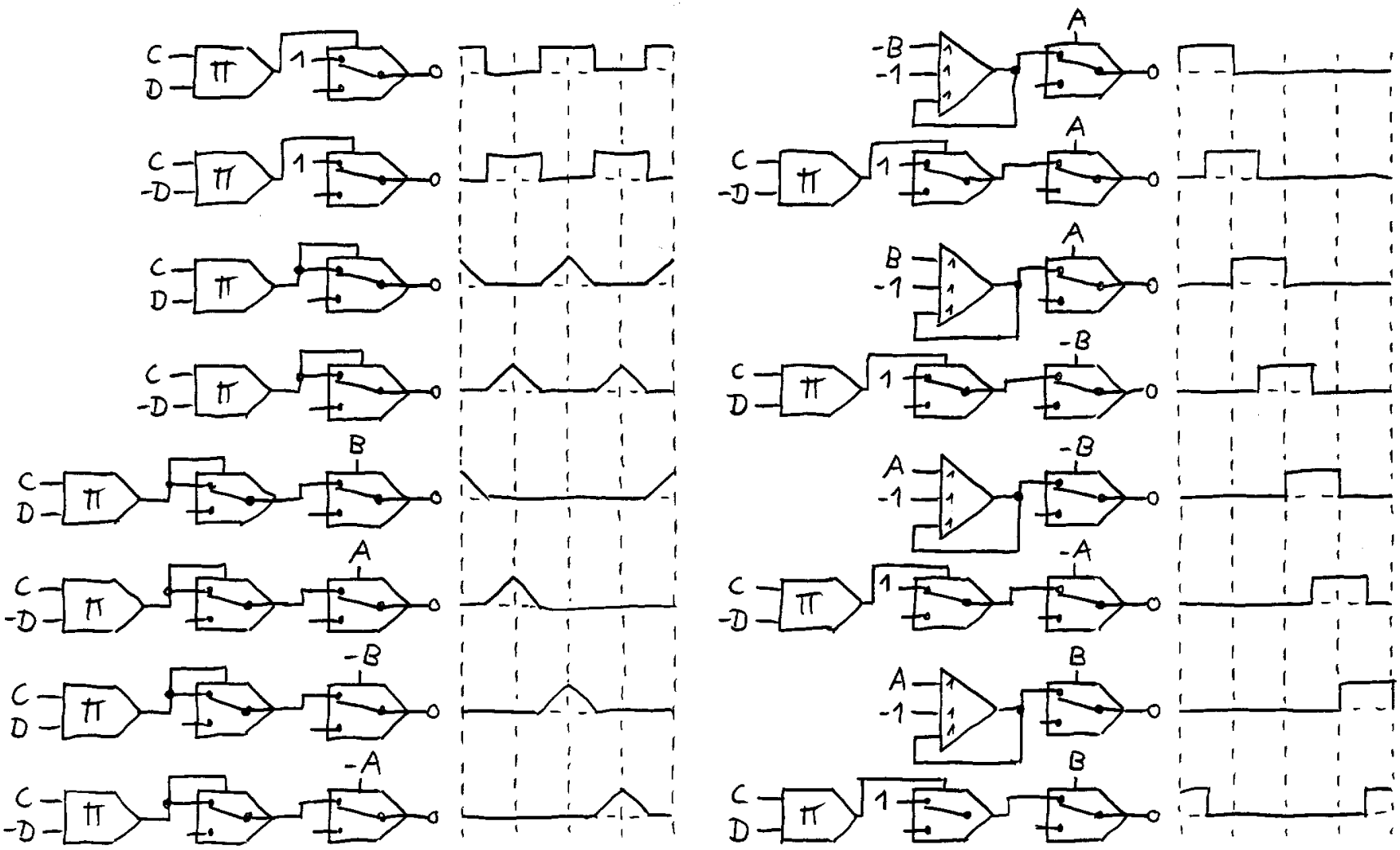



C
D
D
C
-D
C
-D
-D
C
D
-C
C
-D
-C


C
D
C
C
-D
-D
C
D
-C
C
-D
D


A
C
A
-A
D
A
B
D
-B
B
C
-B
A
C
-A
-A
D
-A
B
D
B
B
C
B

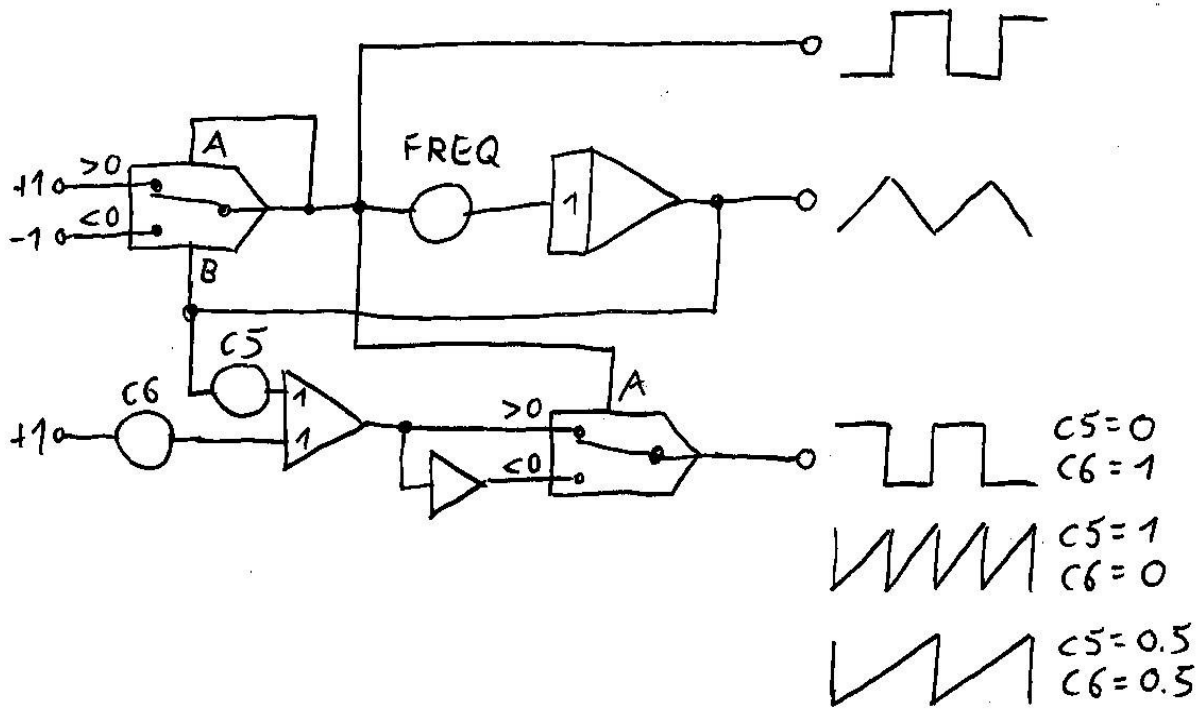

C
C
-D
-D
-C
-C
D
D
A
D
A
B
-D
A
B
-C
A
A
-C
A
-A
D
B
-D
-A
B
-C
-A
A
-C
-A


C
D
1
C
-D
1
C
D
C
-D
B
C
D
A
C
-D
-B
C
D
-A
C
-D
-B
A
-1
A
C
-D
1
A
B
-1
-B
C
D
1
A
-1
-B
-A
C
-D
1
A
-1
B
B
C
D
1

5.18 Sawtooth Generator

The upper half of this circuit is the well known triangle / square wave generator from figure 5.13 in Bernd's book. The output of the lower half depends on the setting of coefficients C5 and C6, and can be a square wave, or a sawtooth wave with the same or with double frequency.

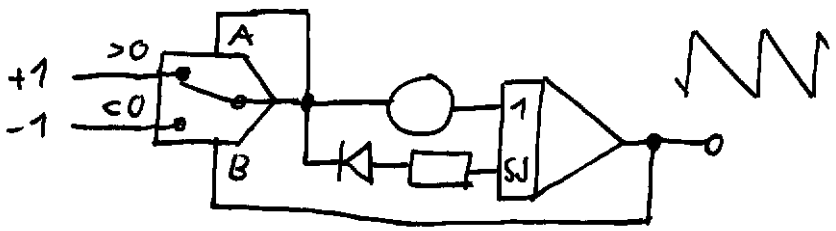
If you want falling sawtooth instead of rising sawtooth, you can swap the inputs of the second comparator.



A sawtooth wave can also be generated with this circuit:

The resistor should not be smaller than about 330 kΩ (or 10 kΩ if the integrator is in SLOW mode). Otherwise the ramp would be too fast for the comparator, so that the output signal would overshoot.

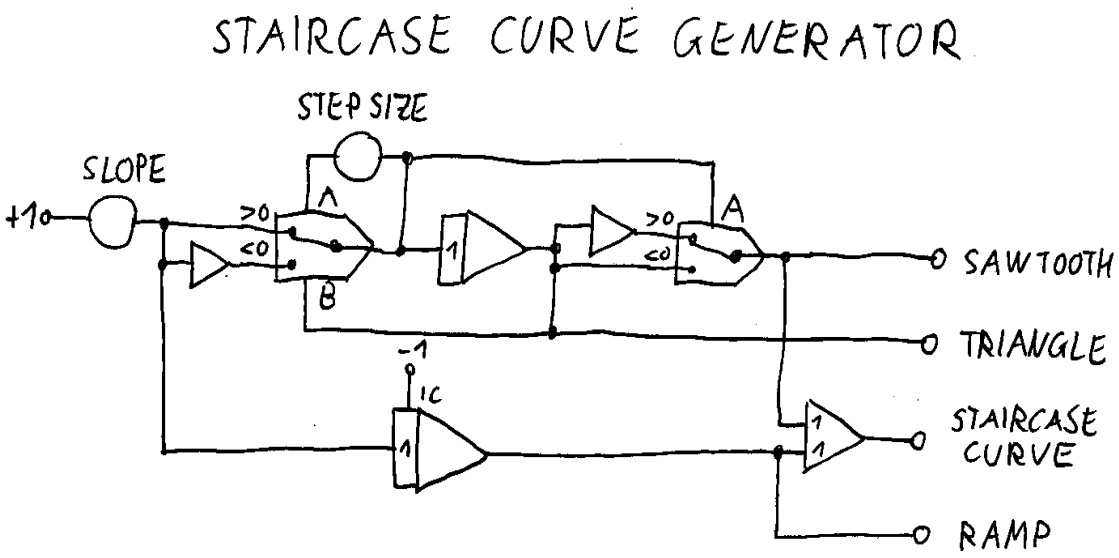
For a rising sawtooth, reverse the polarity of the diode.

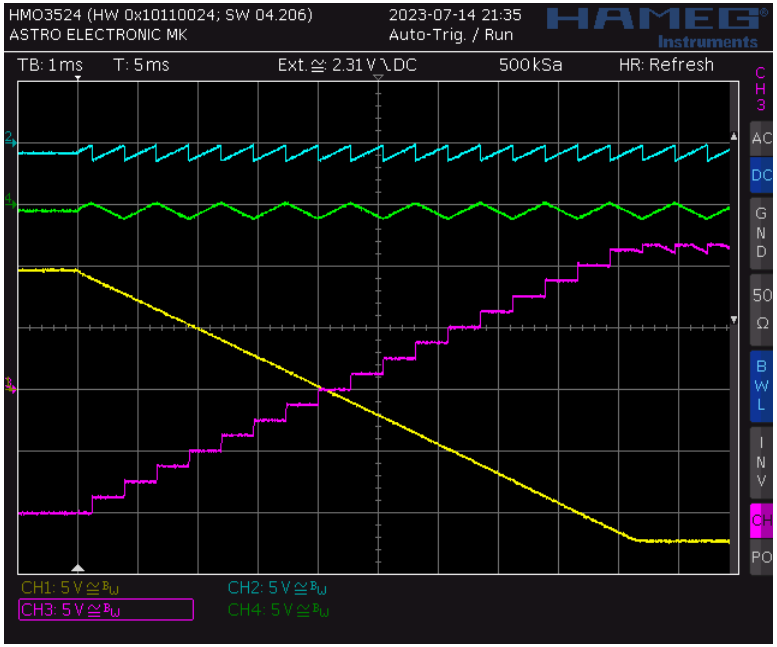


5.19 Staircase Curve Generator

A staircase curve is simply the sum of a ramp signal and a sawtooth signal, which both have the same slope but with opposite sign.

Make sure that the two integrators have the same time constant.

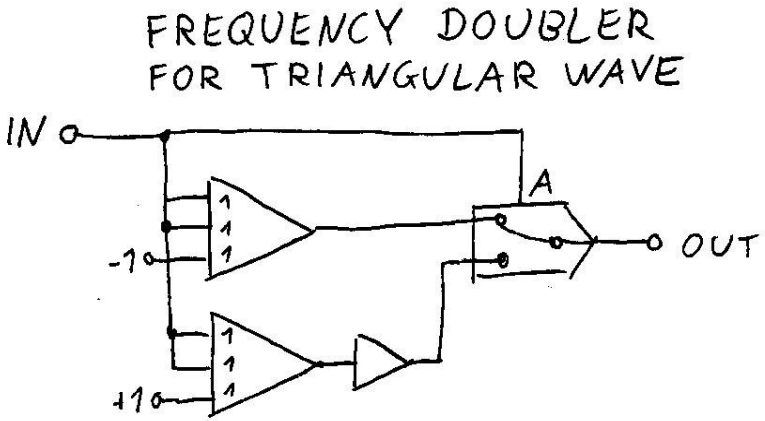


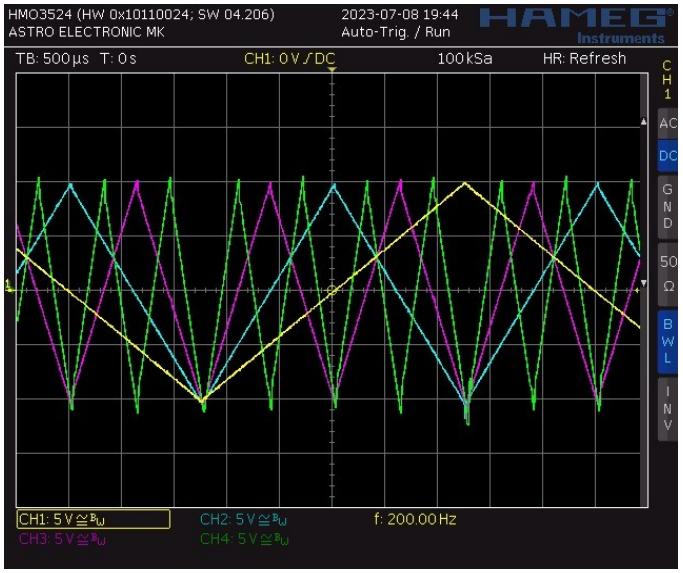


Note: It might be required to give the upper integrator a small non-zero initial condition.

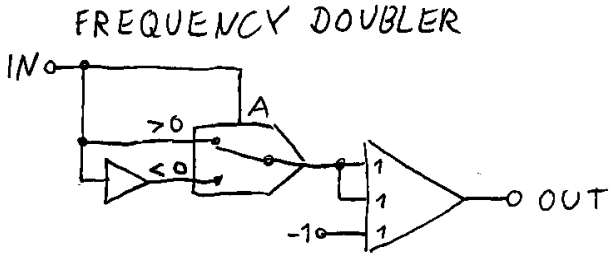
5.20 Frequency Doubling for Triangular Waves

This is a frequency doubler for triangular waves. It does only work if the input signal has the correct amplitude from -1 to +1. The output can be cascaded to the input of the next doubler stage. In the oscilloscope screenshot yellow is the input signal, and cyan, magenta and green are cascaded outputs. This example runs on two THATs.

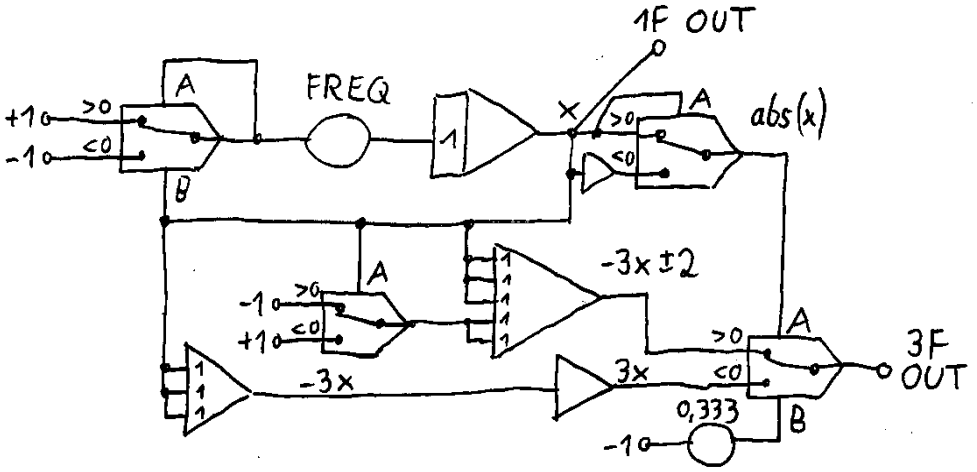


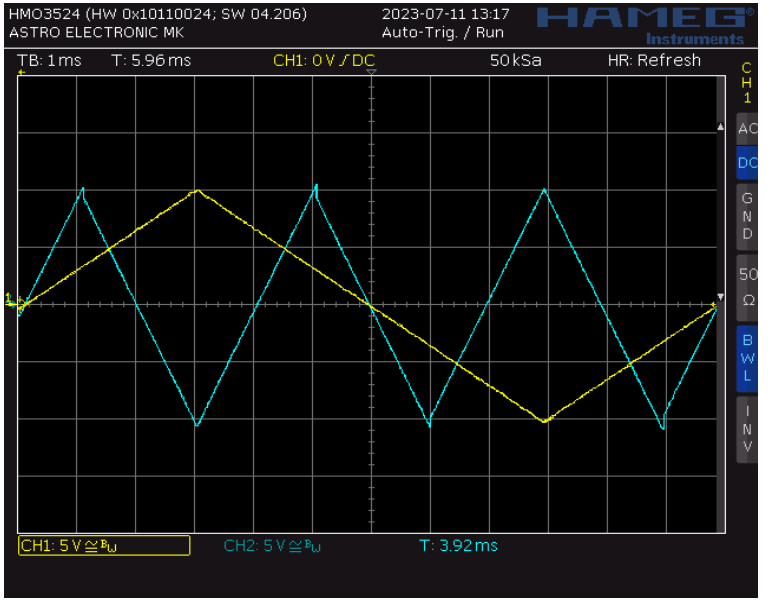


This circuit is simpler and has the same function:

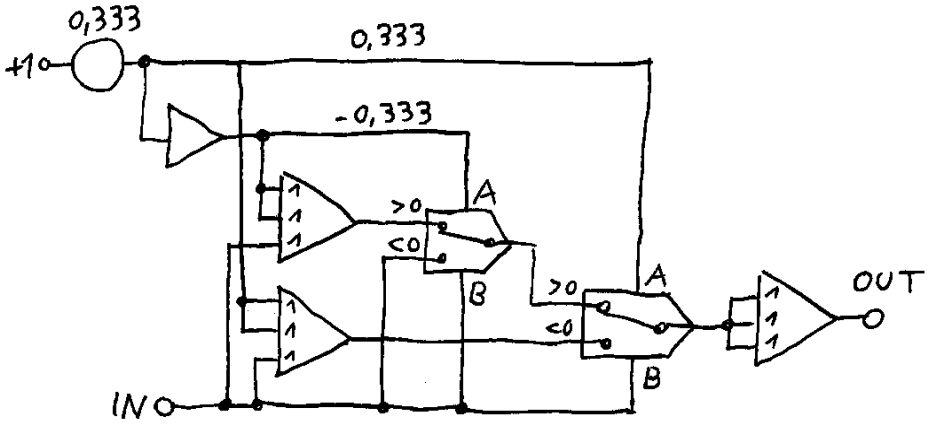


5.21 Frequency Tripling for Triangular Waves

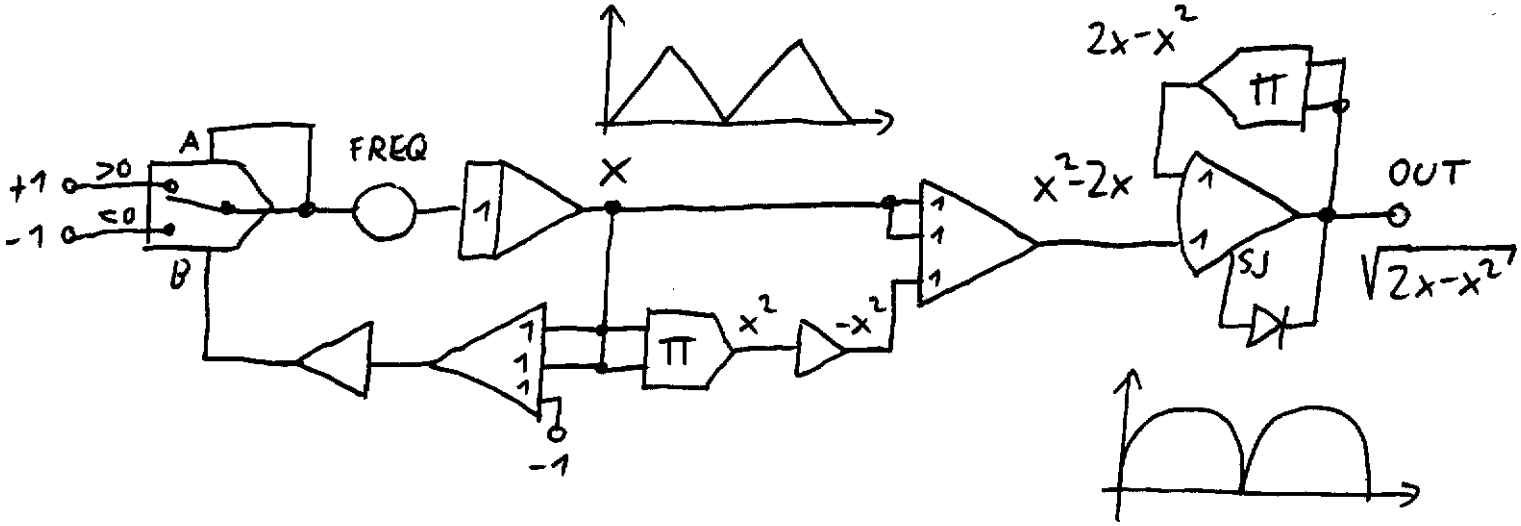


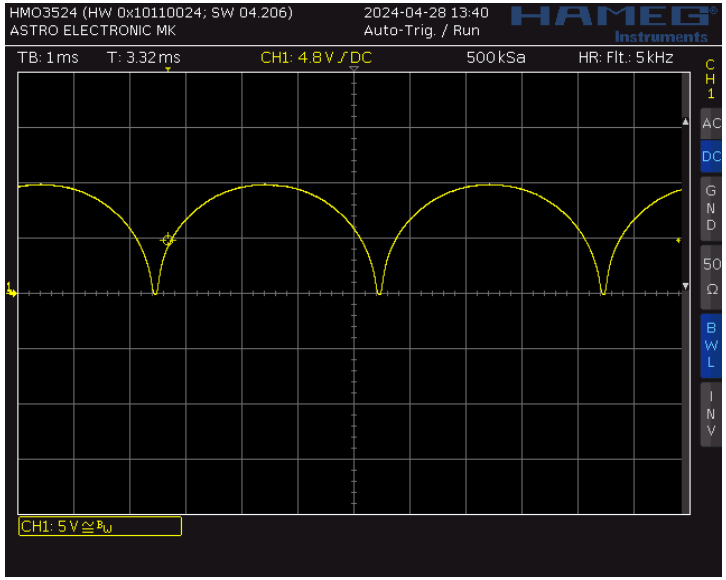


This frequency tripling circuit is better (output is inverted):



5.22 Rectified Semicircle Wave Generator

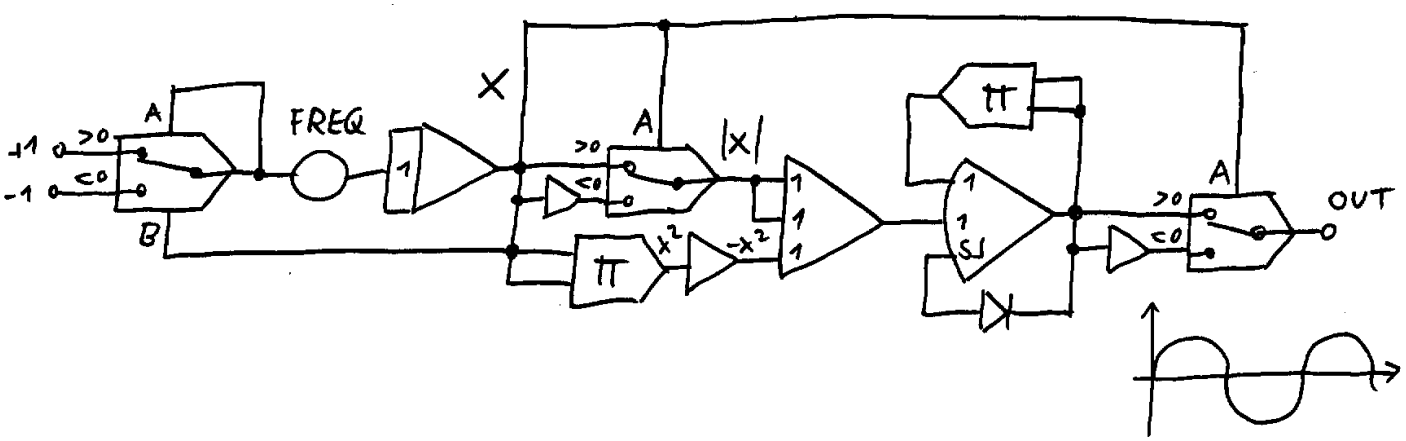


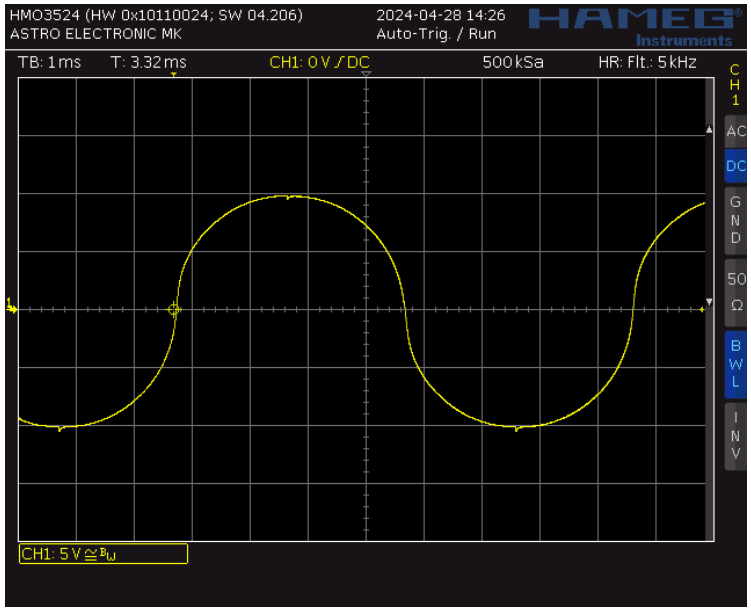


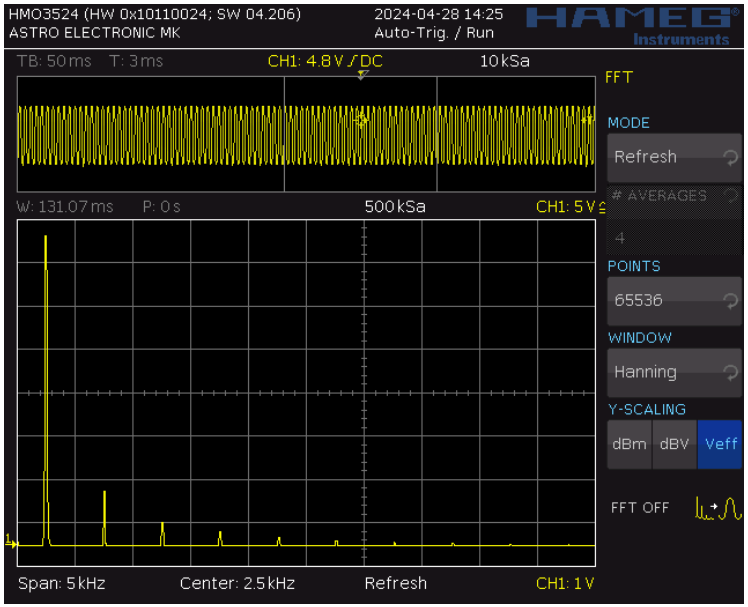
5.23 Semicircle Wave Generator

This is the non-rectified version of the semicircle wave generator. It requires 3 comparators.

The signal doesn't contain many harmonics.



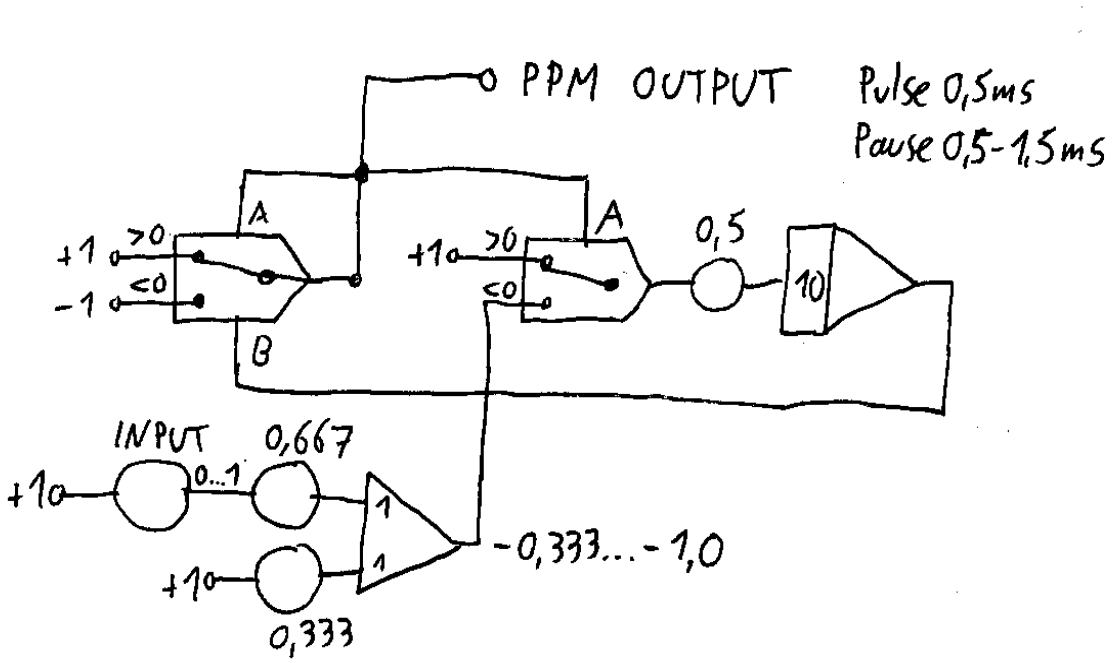




5.24 Pulse Position Modulation

Simple example for pulse position (or pulse pause) modulation. It has a fixed 0.5ms pulse width and a variable 0.5 ms - 1.5 ms pause. But the pause width is a nonlinear function of input voltage. Who has a better idea to make it linear?

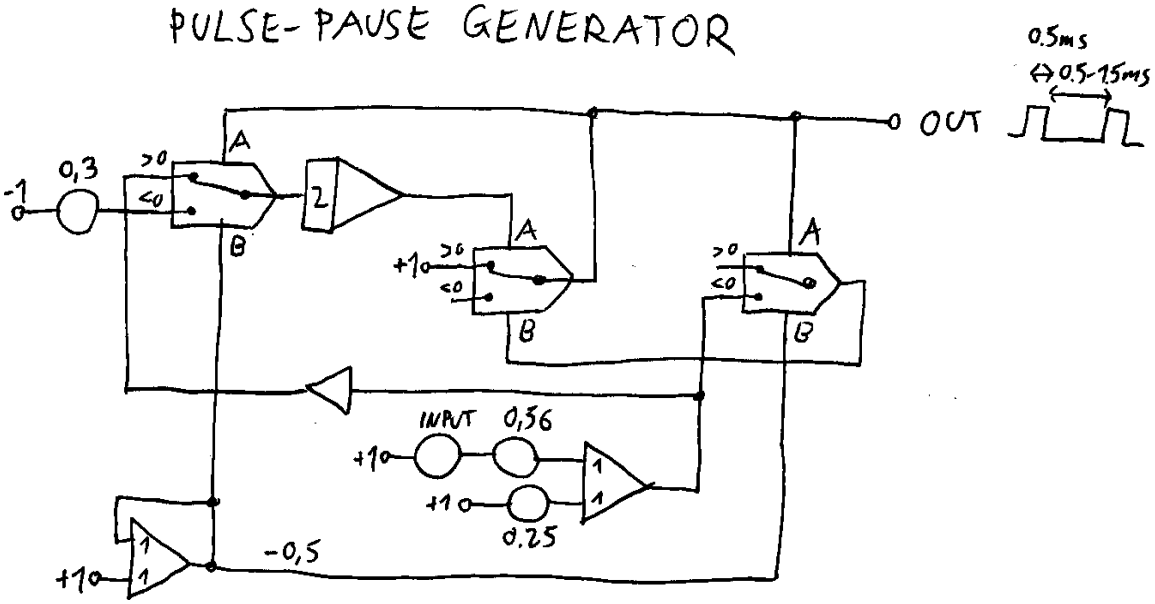
We need a voltage to time converter. The problem is that this circuit is a voltage to slope converter. Slope is voltage / time. That's why in this circuit the voltage is proportional to 1/time. I think the trick is to charge the capacitor with a constant slope to a variable voltage. Then time is proportional to voltage. But how can the capacitor (with variable voltage) be discharged in constant 0.5ms?



This is the solution. The pause duration does linearly depend on the input signal.

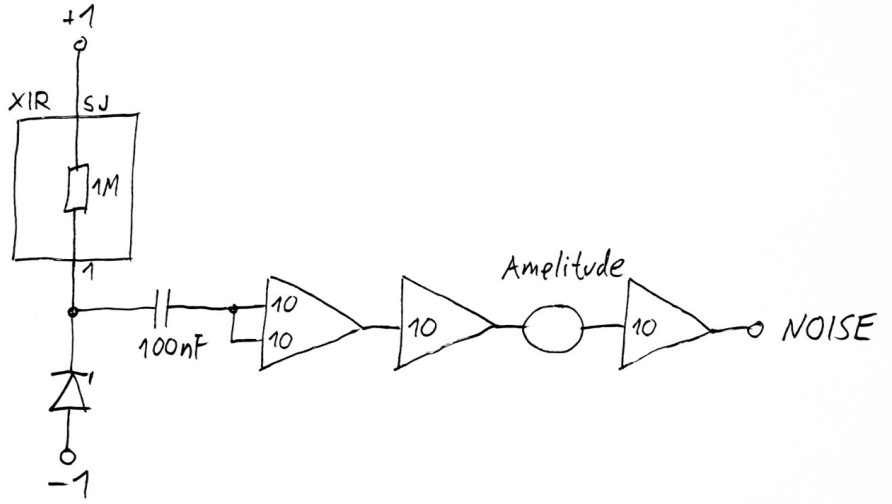
How does it work?

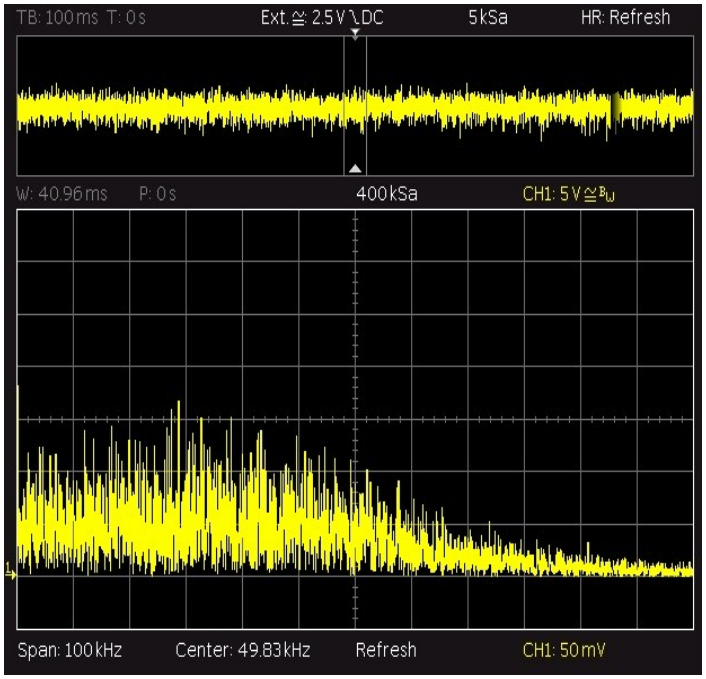
- **The variable pause is created by integrating with a constant slope (-0.3) from zero to a variable peak level, which depends on the input signal.**
- **The fixed pulse duration is created by integrating with a variable slope (depending on the input signal) from the variable peak down to zero. The combination of variable peak level and variable slope gives a fixed duration.**
- **The left comparator selects the slope for integration.**
- **The middle comparator selects the digital output level. The +1 input signal can be changed to other levels, if at the same time the -0.5 signal is changed to -0.5 · output level.**
- **The right comparator selects the compare level for determining the end of the integration slope.**



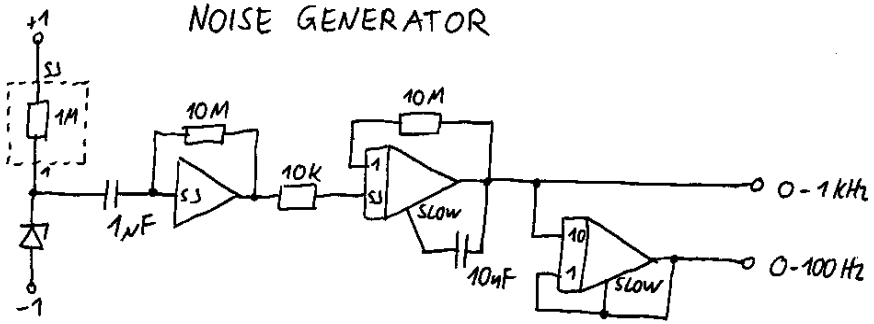
5.25 Noise Generators

This is a noise generator. The oscilloscope screenshot shows voltage over time and the FFT spectrum. The noise signal is usable up to about 50 kHz.

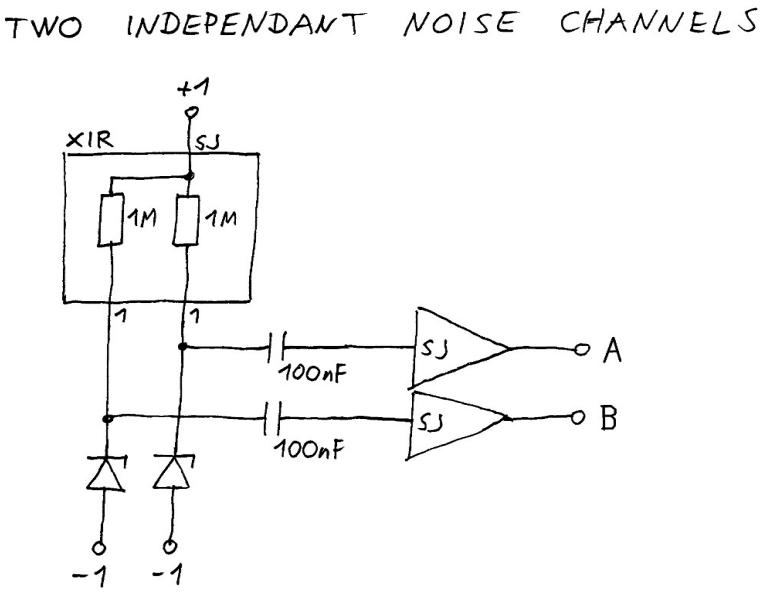


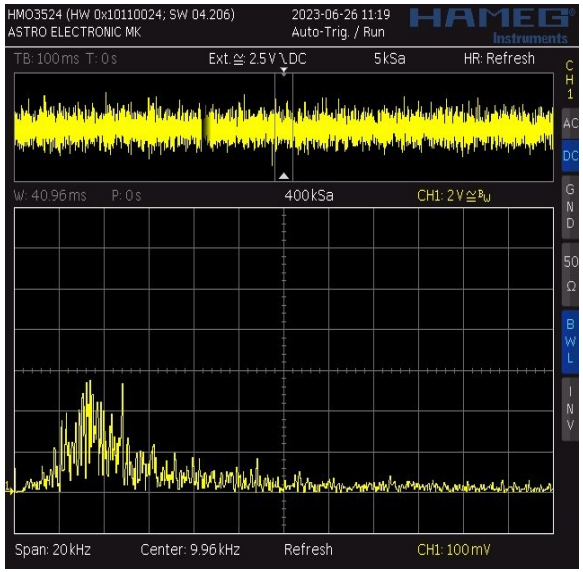


Noise Generator with about 1 kHz and 100 Hz bandwidth:

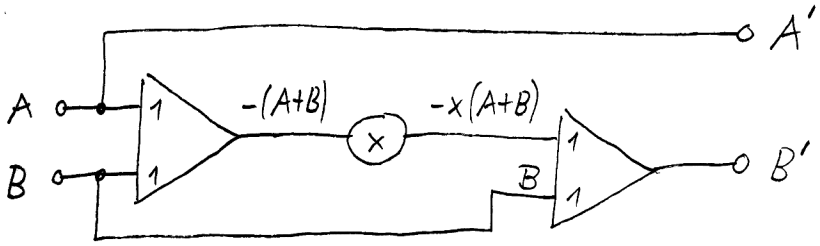


This noise generator is even simpler and has two independant channels. You need only one XIR block, two zener diodes, two capacitors and two inverters. It has a different noise spectrum with a peak at about 3kHz.



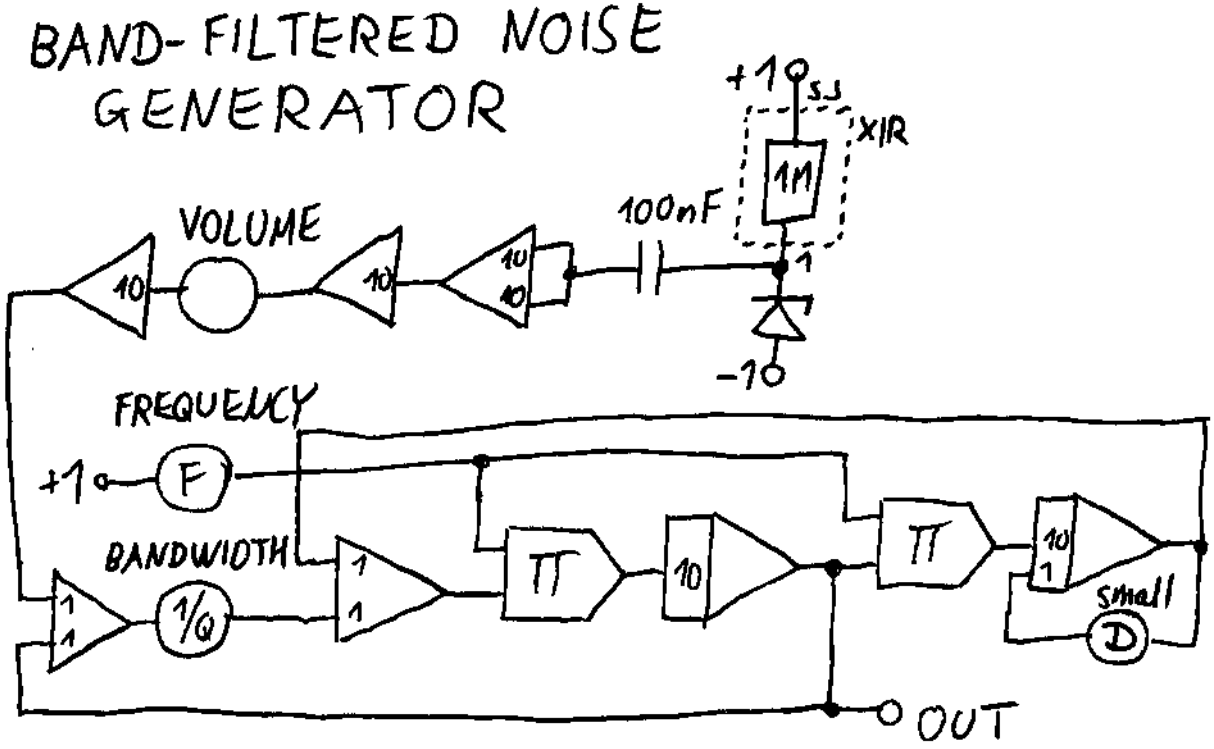


With this circuit the two independant noise signals A and B can be transformed into two signals A' and B' with adjustable degree of correlation x, from 0 to 1. If x = 0, then B' = -B and A' = A, which means the two signals aren't correlated. If x = 1, then A' = A and B' = A = A', which means the signals are identical or 100% correlated.



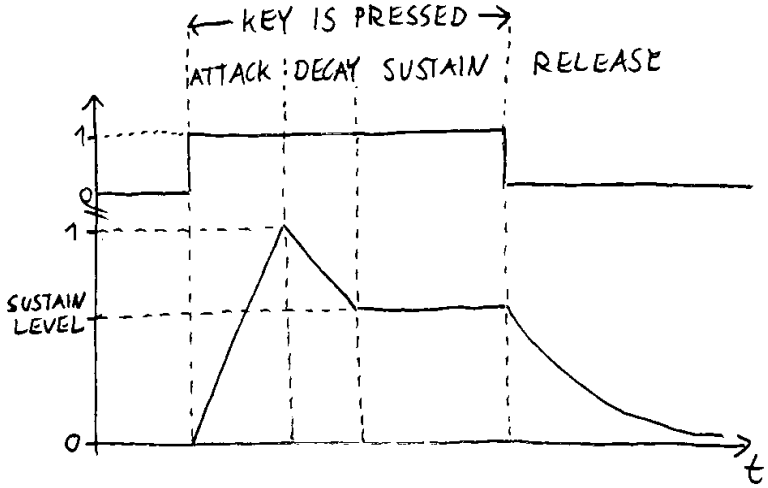
5.26 Band-Filtered Noise Generator

This is a noise generator followed by a bandpass filter with adjustable center frequency (from 0 to 1592 Hz) and quality factor (or relative bandwidth). The filter may oscillate when 1/Q is set very small. In this case you can increase the D coefficient for damping the oscillation. Normally the D coefficient should be zero.



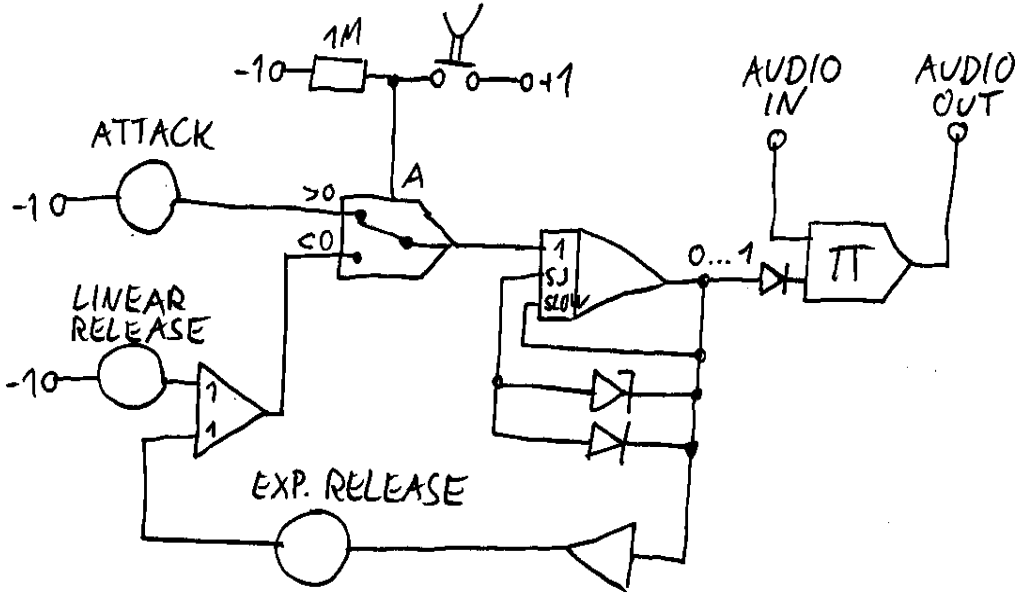
5.27 Envelope Generator for Synthesizer

The envelope consists of four segments Attach - Decay - Sustain - Release:

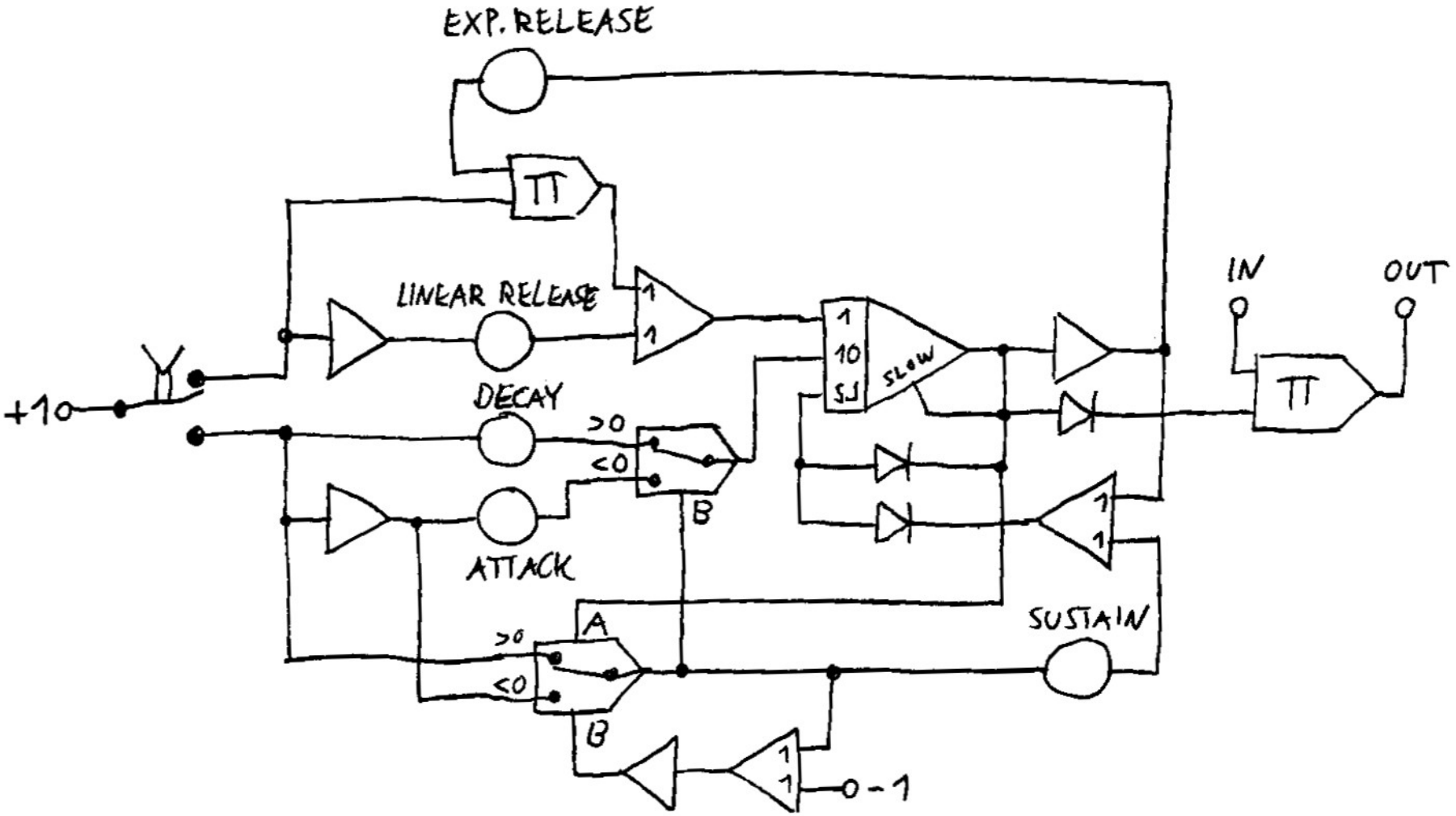


Below is the simplified version with Attack - Sustain - Release:

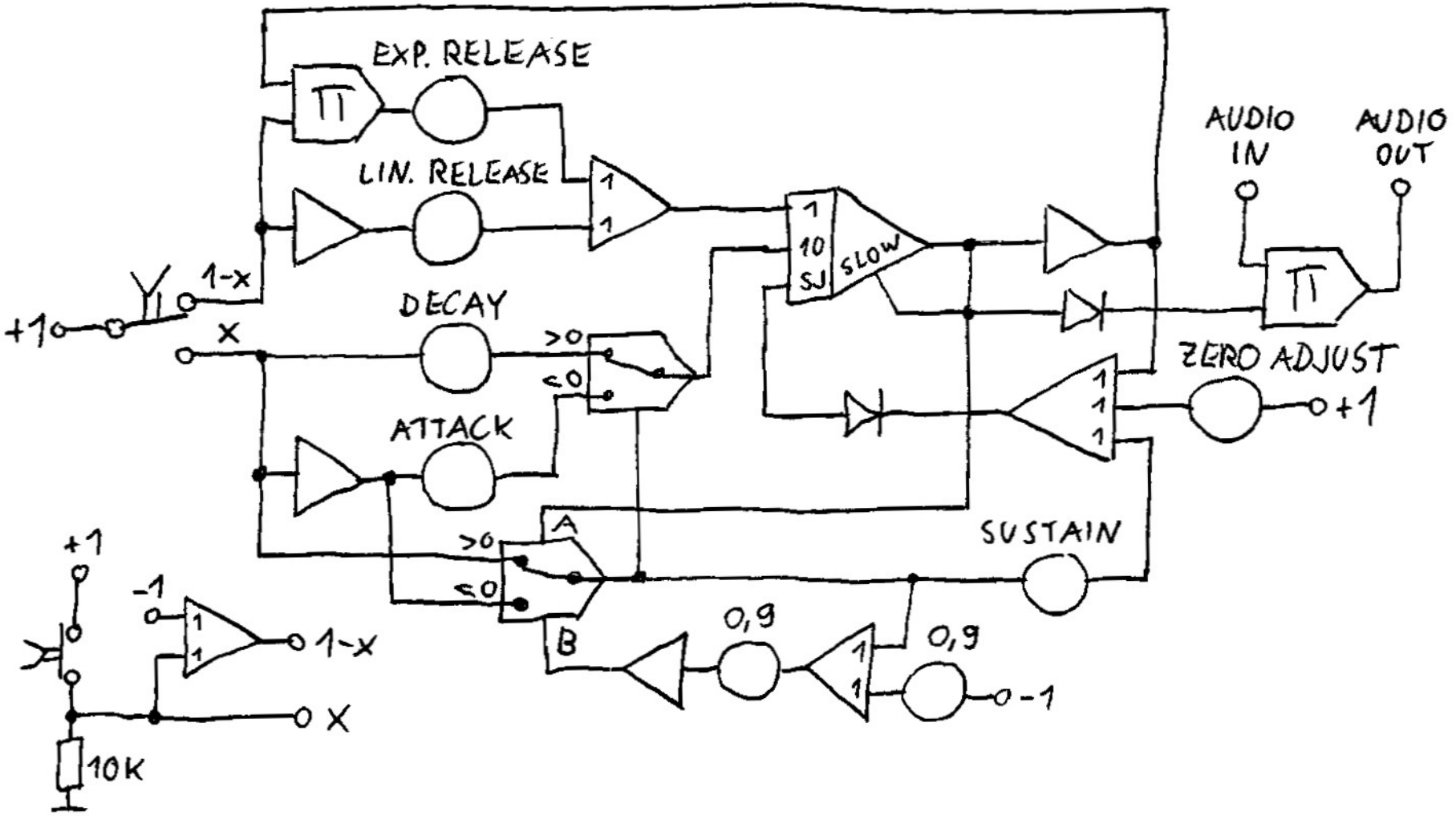
The output of the integrator can go down to about -0.7V, that's the forward voltage of the feedback diode. But we need exactly 0V at the input of the multiplier to make the output silent. That's why there is a diode at the input of the multiplier.



This is the envelope generator with four segments Attach - Decay - Sustain - Release. It doesn't work if you set the sustain level too small.



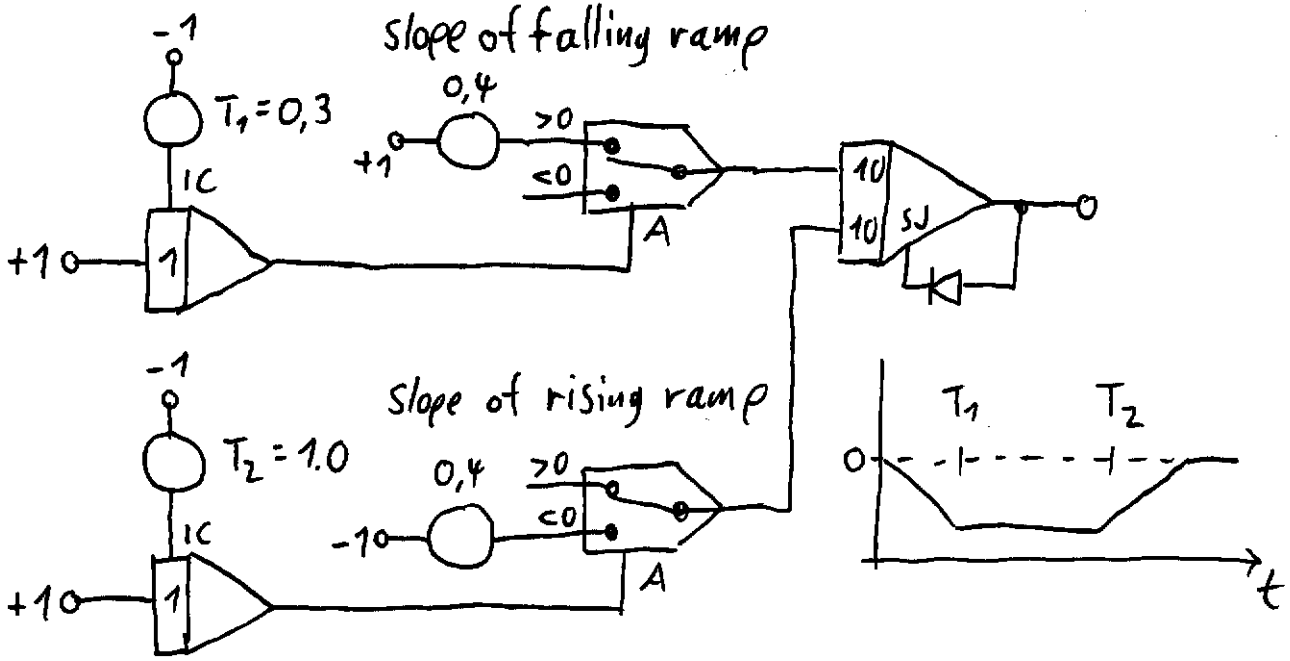
This is a slightly improved version of the previous circuit. The upper feedback diode was removed, because it was unnecessary. The lower diode is doing all the job. When the key is not pressed, the output level can be adjusted to zero with the ZERO ADJUST coefficient. It's best if you make this adjustment in exponential release mode. In the lower right corner is a circuit for a simple pushbutton input.



5.28 Bathtub Curve

See also: https://en.wikipedia.org/wiki/Bathtub_curve

This circuit makes a sequence of falling slope / constant level / rising ramp.
The times T1 and T2 and the slopes of both ramps can be set with coefficients.
The first two integrators are timers. They start with a positive initial value (negative at the IC input, positive at the output), and then the output signal ramps down until the comparators switch when zero level is reached.



5.29 FFplay as Signal Generator

You can create input signals for THAT with the soundcard of a PC. The output voltage is typically about 2.7 V peak-peak. This can for example be done with a batch file which invokes FFplay. This is an example for creating a 1kHz sin/cos quadrature signal:

```
set "F=1000" :: Frequency in Hz
set "V=1" :: Volume
ffplay -f lavfi -i aevalsrc="%V%*sin(2*PI*%F%*t)|%V%*cos(2*PI*%F%*t)":c=stereo:s=48000
pause
```

For more details about FFmpeg and FFplay see also: http://www.astro-electronic.de/FFmpeg_Book.pdf

5.30 Gauss Function Generator

https://analogparadigm.com/downloads/alpaca_42.pdf

5.31 sinc(x) Function Generator

sinc(x) is called the "sine cardinal" function.

https://analogparadigm.com/downloads/alpaca_49.pdf

6 Time and Amplitude Scaling

6.1 Time Scaling

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 77ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

See also: EAI Handbook of Analog Computation (from 1971), page 113ff
https://analogmuseum.org/library/eai_handbook.pdf

See also: Stice, James Edward: Electronic analog computer primer (from 1965), page 67, in this book the time scale factor is called "a" instead of "β".
https://archive.org/details/electronicanalog0000stic

See also: Adler, Helmut: Elektronische Analogrechner (from 1962-1970), page 30ff, in this book the machine time is called "$\bar{t}$" and the time scale factor is called "M_t".

See also: Korn, Granino A. and Korn, Theresa: Electronic Analog and Hybrid Computers, 1972 Edition, page 9, in this book the machine time is called "τ", the problem time is called "t" and the time scale factor is "α_t" instead of "β", where $\alpha_t > 1$ corresponds to a "slow time scale" which means the calculation on the analog computer is slower than the real-world problem.

See also: Laning, J. Halcombe and Battin, Richard H.: Random Processes in Automatic Control, page 374, in this book the machine time is called "τ", the problem time is called "t" and the time scale factor is "a" is defined as a = t / τ, so that "a" is the inverse of "β". If a > 1, the calculation on the analog computer is faster than the real-world problem.

t = Problem time, this is the duration of the real-world problem

τ = Machine time (tau), this is the duration of the simulation on the analog computer $\tau = \beta \cdot t$

Time scale factor: $\beta = \tau / t$

β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems.

β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions.

From EAI Handbook page 115:

Gerneral rule: To speed up or slow down the computer solution by a factor β, every integrator input must be divided by β.

β < 1 speed up the computer solution β > 1 slow down the computer solution

Note:
There are two errors in Bernd Ulmann's book "Analog and Hybrid Computer Programming" on page 57:
In the sentence "Machine time is running faster than problem time ...", replace "β > 1" by "β < 1".
In the sentence "In other cases where the problem time is too short ...", replace "β < 1" by "β > 1".

6.1.1 Reversing Time

A simulation can be changed from forward-in-time to backward-in-time simply by inverting all inputs of all integrators. The initial conditions remain unchanged.

6.2 Amplitude Scaling

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 22ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

See also: EAI Handbook of Analog Computation (from 1971), page 113ff
https://analogmuseum.org/library/eai_handbook.pdf

See also: Stice, James Edward: Electronic analog computer primer (from 1965), page 77, in this book the same thing is called „Magnitude Scaling“.
https://archive.org/details/electronicanalog0000stic

See also: Adler, Helmut: Elektronische Analogrechner (from 1962-1970), page 28ff.

See also: Laning, J. Halcombe and Battin, Richard H.: Random Processes in Automatic Control, chapter C4 on page 372ff

There are two different approaches for amplitude scaling:

1. **Clearly distinguish between *problem variables* and *machine variables*.**
   ***Problem variables* are the variables from the real world. They have a range from a minimum to a maximum. In most cases they also have a physical unit (m, kg, km/h...)**
   ***Machine variables* are scaled so that they don't exceed the working range of the analog computer, which is from -1 to +1. They don't have a physical unit.**
   At first I thought this is the best approach, but then I changed my mind and followed the other approach:
2. **This is my preferred approch. Each machine variable has a physical unit attached to it. The unit may also include a number, for example [30 m/s]. All my examples in the next chapters (beginning with "Rules for Amplitude and Time Scaling") are using this approach, because I think it's much simpler.**

The rest of this chapter does only apply to the first approch. It's irrelevant for the (preferred) second approach.

A simple example for amplitude scaling can be found in: "A chaotic Sprott system" https://analogparadigm.com/downloads/alpaca_43.pdf

We must distinguish between two types of variables:

In the above paper the equations are given with problem variables x, y, z, x', y' and z':

$x' = y \cdot z$

$y' = x - y$
$z' = 1 - x \cdot y$

In this case it's known that the values of all variables fit in the -10 to +10 range. That means we can divide them by 10 to get the machine variables, which are in the -1 to +1 range.

It's confusing if the same names are used for problem variables and machine variables. It's better to use different names.
For example the machine variables can be indicated by a bar ($\overline{x}$) above the variable, or they can be given an index (x_M).

Problem variable in range [-10..+10]	Machine variable in range [-1..+1]	Conversion formulas	
x	$\overline{x}$	$\overline{x} = x / 10$	$x = 10 \cdot \overline{x}$
y	$\overline{y}$	$\overline{y} = y / 10$	$y = 10 \cdot \overline{y}$
z	$\overline{z}$	$\overline{z} = z / 10$	$z = 10 \cdot \overline{z}$
x'	$\overline{x'}$	$\overline{x'} = x' / 10$	$x' = 10 \cdot \overline{x'}$
y'	$\overline{y'}$	$\overline{y'} = y' / 10$	$y' = 10 \cdot \overline{y'}$
z'	$\overline{z'}$	$\overline{z'} = z' / 10$	$z' = 10 \cdot \overline{z'}$

Note: In this example all variables have the same scaling factor 10. In other examples different variables could have different scaling factors. For example x could have a scaling factor 10 and y could have a scaling factor 50.

Now we can substitute all variables from the given equations by their corresponding machine variables:

$x' = y \cdot z$ ==> $10 \cdot \overline{x'} = 10 \cdot \overline{y} \cdot 10 \cdot \overline{z}$

$y' = x - y$ ==> $10 \cdot \overline{y'} = 10 \cdot \overline{x} - 10 \cdot \overline{y}$

$z' = 1 - x \cdot y$ ==> $10 \cdot \overline{z'} = 1 - (10 \cdot \overline{x} \cdot 10 \cdot \overline{y})$

Finally all three equations can be divided by 10, and we get this set of equations with machine variables:

$\overline{x'} = 10 \cdot \overline{y} \cdot \overline{z}$

$\overline{y'} = \overline{x} - \overline{y}$

$\overline{z'} = 0.1 - 10 \cdot \overline{x} \cdot \overline{y}$

When the problem has been solved on the analog computer, the results can be converted back to problem variables, using the same conversion

formulas.

6.3 Rules for Scaling (The most important Chapter in this Book!)

Time and amplitude scaling is the most complicated thing in analog computing. The idea is to simplify the scaling process by applying a set of rules, step by step.

Scaling Rule 1:

As the first step, describe the problem with SI units (m, s, N, kg, V, A, W ...). Write down the equations.

If already known, write down the desired time scale factor β.
β < 1 means the calculation on the analog computer is faster than the real-world problem, for example population dynamics or astronomical problems.
β = 1 means the simulation is running in realtime.
β > 1 means the calculation on the analog computer is slower than the real-world problem, for example fast chemical reactions or electronic circuits.
If the time scale factor isn't yet known, begin with a reasonable assumption.

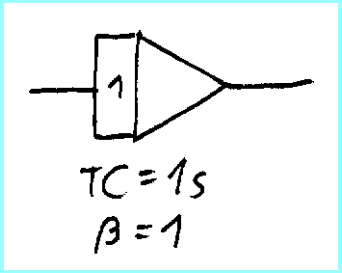
The argument of a function like exp(), log(), sin(), cos() or tan() does never have a unit. If it has a unit (different from 1), then the equation is wrong. Multiply (or divide) the argument by a suitable factor to remove the unit.

Examples:

$y = \exp(t\ [s])$	This is wrong. The argument of the exponential function can't have the unit [s].
$y = \exp(t\ [s] / \tau\ [s])$	This is correct. Dividing the time t by a time constant τ removes the unit from the argument.
$y = \sin(t\ [s])$	This is wrong. The argument of the sine function can't have the unit [s].
$y = \sin(\omega\ [s^{-1}] \cdot t\ [s])$	This is correct, because $\omega = 2 \cdot \pi \cdot f$ has the unit [s-1], which compensates the unit [s] of the time t.

Physical Quantity	Symbols	Important formulas	SI Unit	Symbol	SI unit conversions
Force	F	$F = m \cdot a$ $F = W / s$	Newton	N	$N = kg \cdot m\ s^{-2}$ $N = J / m$ $N = W \cdot s / m$
Mass	m		Kilogramm	kg	$kg = N \cdot s^2\ m^{-1}$
Position, Distance	s, x		Meter	m	$m = N \cdot s^2\ kg^{-1}$ $m = J / N$ $m = W \cdot s / N$
Velocity	v	$v = ds / dt$	Meter / Second	m/s	1 m/s = 3.6 km/h
Acceleration	a, g	$a = dv / dt$	Meter / Second^2	m/s^2	Gravitational acceleration: g = 9.81 m/s^2
Time	t		Second	s	$s = N \cdot m / W$ $s^2 = kg \cdot m / N$ $s = J / W$
Power	P	$P = W / t$ $P = U \cdot I$	Watt	W	$W = V \cdot A$ $W = N \cdot m / s$ $W = J / s$
Energy = Work	E, W	$W = P \cdot t$ $W = F \cdot s$	Joule = Wattsecond = Newtonmeter	J	$J = Ws = Nm = VAs = CV = kg\ m^2\ s^{-2}$ 1 Wh = 3600 J
Torque	M	$M = F \cdot x$	Newtonmeter	Nm	$Nm = kg\ m^2\ s^{-2}$
Pressure	p, P	$p = F / x^2$	Pascal	Pa	$Pa = N / m^2 = J / m^3 = kg\ m^{-1}\ s^{-2}$ 1 Pa = 10^{-5} bar
Electric charge	Q	$Q = I \cdot t$	Coulomb	C	C = As
Voltage	U	$U = R \cdot I = P / I$	Volt	V	$V = W / A = kg\ m^2\ s^{-3}\ A^{-1}$
Current	I	$I = U / R = P / U$	Ampere	A	A = W / V
Resistance	R	$R = U / I$	Ohm	Ω	$\Omega = V / A = kg\ m^2\ s^{-3}\ A^{-2}$
Capacitance	C	$C = Q / V$ $I = C \cdot dU / dt$	Farad	F	$F = A^2\ s^4\ kg^{-1}\ m^{-2}$
Inductance	L	$U = L \cdot dI / dt$	Henry	H	$H = kg\ m^2\ s^{-2}\ A^{-2}$

Scaling Rule 2:



Draw the schematic diagram for the analog computer, under the assumption that

- **All integrators have a time constant TC = 1s**
- **All integrators have a time scale factor β = 1**

Write "TC = 1s" and "β = 1" below each integrator.

All signals and coefficients should have a physical unit attached to them, which is written in [ ] brackets.

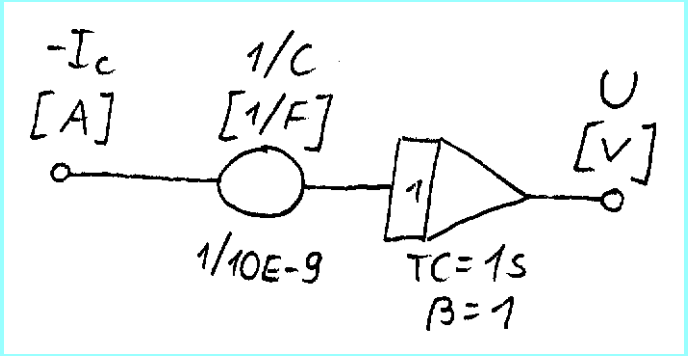
Keep in mind that all integrators and summers are inverting.

The resulting schematic diagram should be mathematically correct, but it won't run on the analog computer because it still has many problems:

- **The time scale factor may be different from the desired value.**
- **The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode).**

- **The coefficients may be larger than +1 or smaller than -1 or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.05 and 1. In some cases 0.01 may be acceptable as well.**

Example:



Given is the formula for voltage dU and current I_C at a 10 nF capacitor:

$dU / dt = I_C / C$ C = 10 nF

$-I_C$ and U have different signs because the integrator is inverting.
It doesn't care if the minus sign is at the input or at the output.
You could also write I_C and -U.

What about the integrator's time constant k_0?
I did prefer not to use k_0 because I didn't find a definition for it. It was unclear if the unit of k_0 is [s] or [1/s] or if it's a dimensionless factor.
Update: Later I found a definition for k_0 in the book "Analogrechnen" by Giloi and Lauber (on the page before page 1, and on page 65):

Scaling Rule 3:

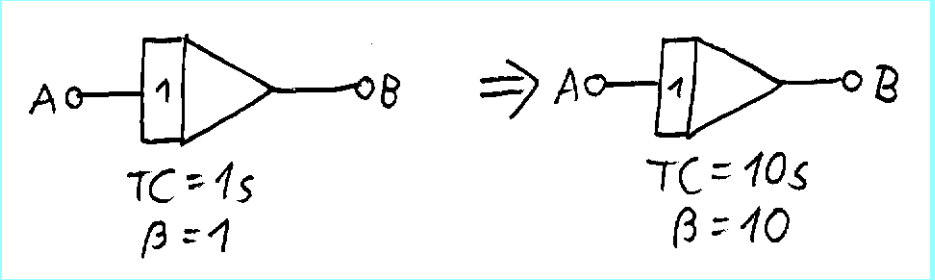
Change the time scale factor β to the desired value. To do this, multiply the time scale factors β of <u>all</u> integrators by a suitable factor X and multiply the time constants TC of <u>all</u> integrators by the same factor X.

If the circuit shall run in realtime (β=1), then this step is unnecessary.

Write the new time scale factors β and time constants TC below each integrator.

All integrators in the circuit must have the same time scale factor β.

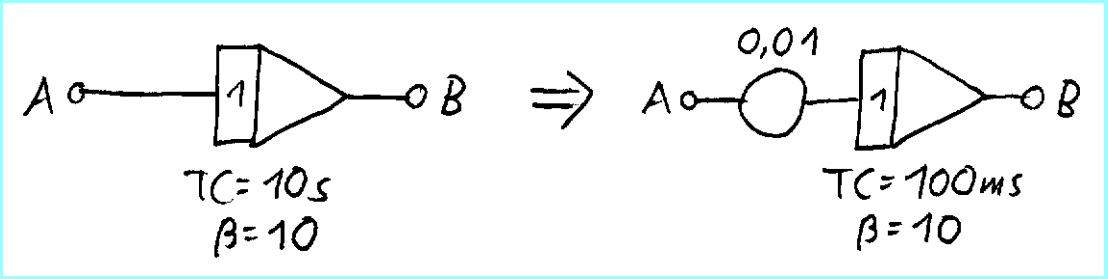
If the circuit contains delay lines, these must be changed as well.



Scaling Rule 4:

The time constant TC of an integrator can be multiplied by a factor x, if all inputs of this integrator are multiplied by the same factor x. This operation doesn't change the time scale factor β and the initial condition of the integrator. Each integrator can have its own factor x.

Choose x so that the time constant of the integrator becomes either 1 ms or 100 ms, because these time costants are available in THAT (1 ms in normal mode, and 100 ms in SLOW mode). In the example to the right x is 0.01.

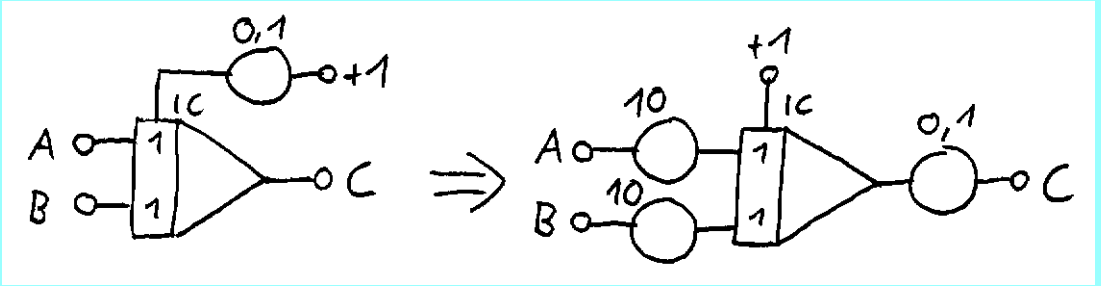


The integrators may have different time constants TC, but they must all share the same time scale factor β.
With other words: The time constant TC is a property of each integrator, however the time scale factor β is a property of the whole circuit.

Scaling Rule 5:

If all input signals of an integrator (or summer) are multiplied by a factor x, then the output of the integrator (or summer) does also become larger by factor x. This introduces an error that must be compensated by inserting a 1/x coefficient at the output of the integrator (or summer). In the example to the right the factor x is 10.

If the integrator's initial condition (IC) is different from 0, then it must also be multiplied by factor x.

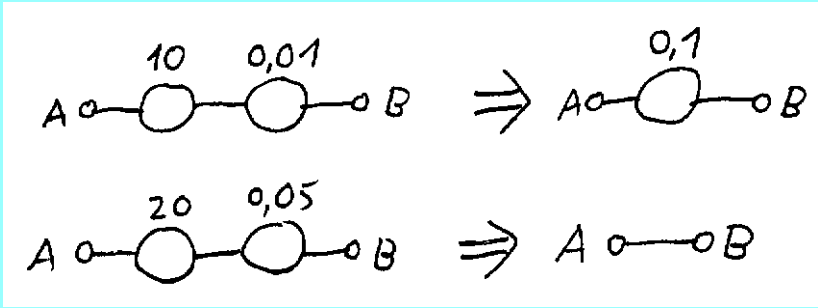


This operation doesn't change the integrator's time constant TC and time scale factor β.

Scaling Rule 6:

Multiple coefficients in series can be combined into a single coefficient, which is the

If the coefficient is 1, it can be removed.



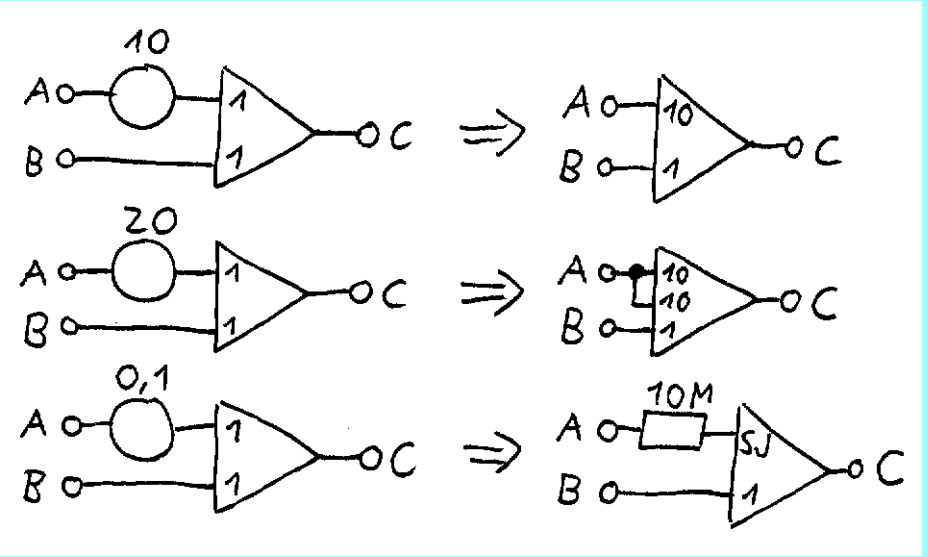
Scaling Rule 7:

If a coefficient at the input of a summer or integrator is larger than 1, it can be removed if the signal is connected to an input with a correspondingly higher weight.

For example a coefficient with value 10 is equivalent to connecting the signal to an input with weight 10, or a coefficient with value 20 is equivalent to connecting the signal to two inputs with weight 10 each.

Inputs with weights smaller than 1 can also be realized by connecting the signal through a resistor to the summing junction of a summer or integrator.

Use 10 MΩ for weight 0.1, or 100 MΩ for weight 0.01 or 1 GΩ for weight 0.001.



Scaling Rule 8: **Very important!**

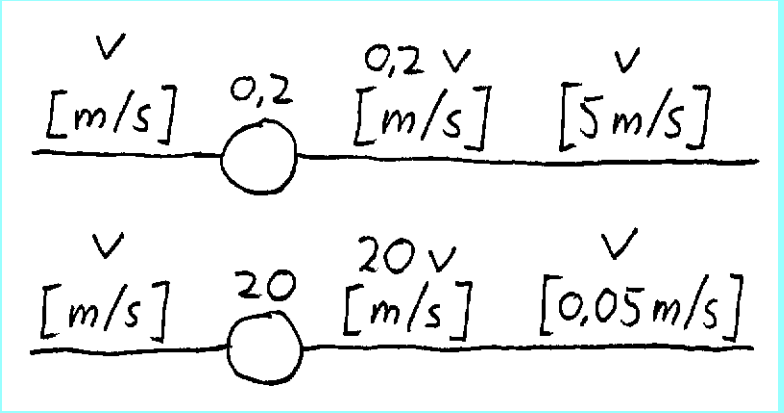
If an input signal is sent through a coefficient, the output signal can be described in two different ways:

- **As a different problem variable with the same unit,**
  in this case the problem variable is multiplied by the coefficient value.

- **As the same problem variable with a different unit,**
  in this case the output unit is divided by the coefficient value.

  Output_Unit = Input_Unit / Coefficient

  If the coefficient is smaller than 1, then the output unit becomes larger.
  If the coefficient is larger than 1, then the output unit becomes smaller.



Each problem variable (here: v or 0.2v or 20v) equals the product of the machine variable (-1 to +1) and the unit in [ ] brackets.

The unit in [ ] brackets is equivalent to 1 machine unit.
For example, the unit [10m/s] means if the machine variable is 0.5 machine units, then the problem variable is 5m/s.

You can multiply both the problem variable (here: 0.2v) and its unit (here: [m/s]) by the same factor (here: 5), and the resulting signal is still the same as before.

Example:

Let's assume the input machine variable is 0.5 and represents a velocity v with unit [m/s]. The input problem variable is v = 0.5 [m/s].
The coefficient is set to 0.2. The output signal can be described in either of these two ways:

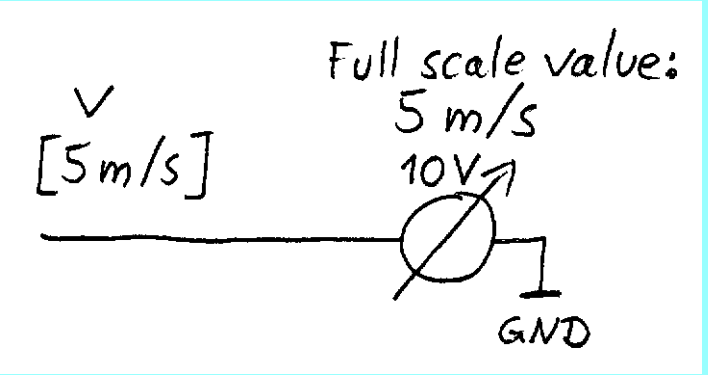
- **Velocity 0.2v with unit [m/s], in this case the output problem variable is 0.2v = 0.5 · 0.2 [m/s] = 0.1 [m/s], or v = 0.5 [m/s].**
  The problem variables at the input (v) and at the output (0.2v) are different, but the value of v is still the same.

- **Velocity v with unit [5 m/s], in this case the output problem variable is v = 0.5 · 0.2 [5 m/s] = 0.1 [5 m/s] = 0.5 [m/s].**

Scaling Rule 9:

If you connect a signal to an analog 10V meter, the unit in [ ] brackets becomes the full scale value of the voltmeter.

In this example the voltmeter points to its full scale value if the velocity v equals 5 m/s.

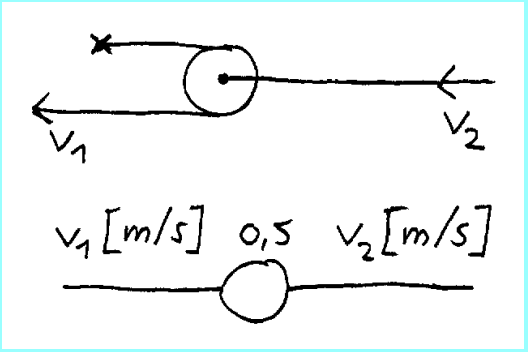


Scaling Rule 10:

If one problem variable is multiplied by a factor to make another problem variable, the unit in [ ] brackets remains the same as before.
In this example velocities v_1 and v_2 are two different problem variables.

v_1 is the velovity of the end of the rope, while v_2 is the velocity of the pulley.

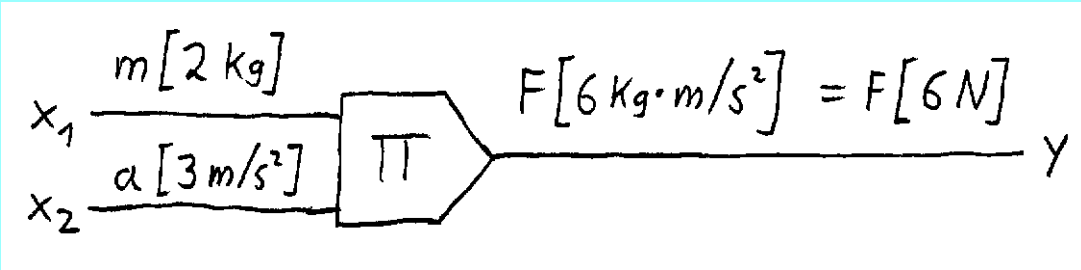
$v_2 = 0.5 \cdot v_1$



Scaling Rule 11:

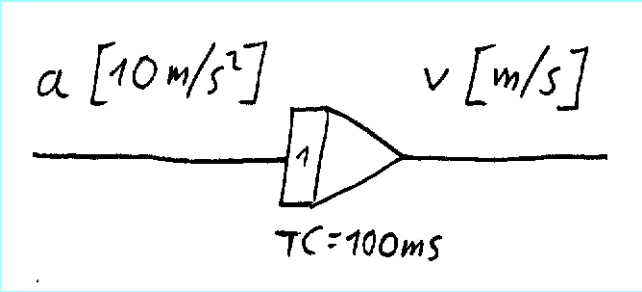
If two problem variables are multiplied, the output unit becomes the product of the units of the two input factors.

Example:
Machine variables: $y = x_1 \cdot x_2$



Scaling Rule 12:

The unit at the output of an integrator equals the unit at its input multiplied by the integrator's time constant TC and divided by the time scale factor β.



Output_Unit = Input_Unit · TC / β

Example (assuming β = 1): 10 m/s² · 0.1s = 1 m/s

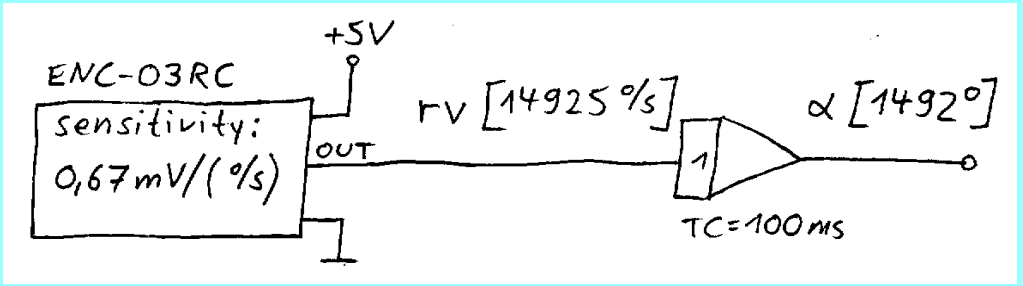
If an integrator input with a different weight w is used, the unit of the output signal becomes:

Output_Unit = Input_Unit · TC / β / w

Scaling Rule 13:

How to find the correct unit of a problem variable from a sensor:

Example: A gyroscope sensor (rotational velocity) has a



Don't confuse the sensitivity S of a sensor with the

To find the unit of the problem variable, you must answer this question:

0.67mV is equivalent to 1°/s rotational velocity.
10V = 1MU is equivalent to 14925°/s rotational velocity, because 10V / S = 14925°/s.
The unit of the rotational velocity signal is [14925°/s].

Problem_variable_unit = 10V / Sensor_sensitivity

If this rotational velocity signal is connected to the input of an integrator with time constant 100ms (SLOW mode), then the output signal is a rotational position (angle) with unit [14925°/s] · 100ms = [1492°].

Scaling Rule 14:

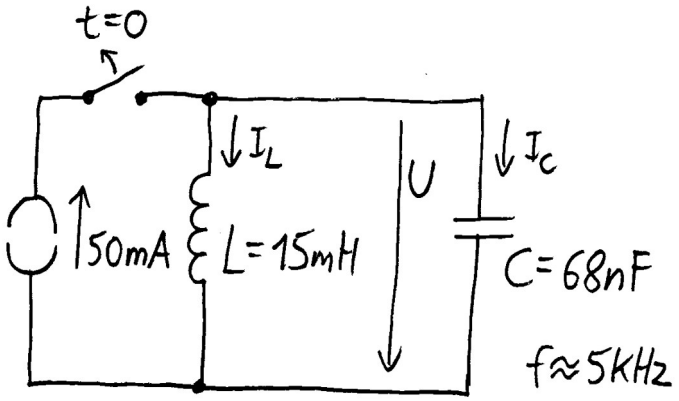
The allowed range for all machine variables is -1 to +1.
Also it should be avoided that coefficients and machine variables become very small (below 0.01 machine units), because the computing elements of the analog computer have offset errors which sum up, and because you can’t set a coefficient precisely to such small values.

The offset error of the TL074H OpAmps used in THAT have a maximum offset error of 4mV and typically 1mV, which is equivalent to 0.0001 machine unit.

If you can't find a configuration where all coefficients and machine variables are in the acceptable range, then you may have to choose a different time scale factor β, or the problem can’t be solved on the analog computer.

7 Examples with Amplitude and/or Time Scaling

7.1 LC Circuit

Rules	Example
Rule 1: **As the first step, describe the problem with SI units (m, s, N, kg, V, A ...). Write down the equations.** **If already known, write down the desired time scale factor β.** **β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems.** **β = 1 means the simulation is running in realtime.** **β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions or electronic circuits.** **If the time scale factor isn't yet known, begin with a reasonable assumption.**	**The current source lets 50 mA flow through the inductor. At t = 0 the switch opens. Simulate the circuit on the analog computer and find out the resonance frequency. In this example we do already know that the frequency is about 5 kHz, which is too fast for THAT. A time scale factor β = 100 seems like a good choice.** **Equations and given data:** **Inductance: $dI_L / dt = U / L$** **Capacitor: $dU / dt = I_C / C$** **Sum of currents: $I_L + I_C = 0$** **Initial condition: $I_L = 50$ mA** **L = 15 mH** **C = 68 nF** 

Rule 2:

Draw the schematic diagram for the analog computer, under the assumption that

- **All integrators have a time constant TC = 1s**
- **All integrators have a time scale factor β = 1**

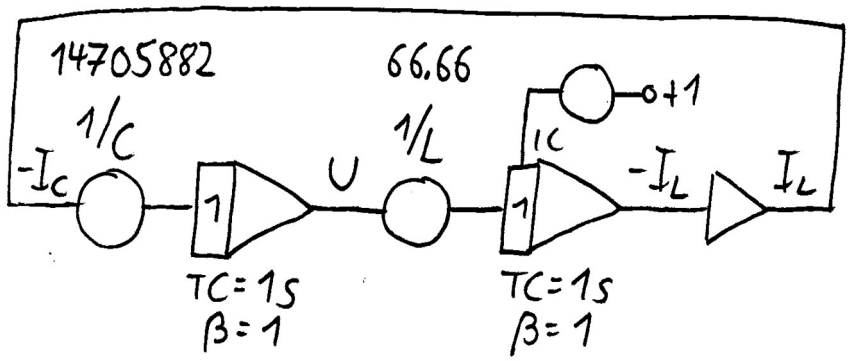
Write "TC = 1s" and "β = 1" below each integrator.

All signals should have a physical unit attached to them, which is normally written in [ ] brackets.

The resulting schematic diagram should be mathematically correct, but it won't run on the analog computer because it still has many problems:

- **The time scale factor may be different from the desired value.**
- **The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode).**
- **The coefficients may be larger than +1 or smaller than -1 or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.05 and 1. In some cases 0.01 may be acceptable as well.**

1 / (68 nF) = 14705882 F^{-1}
1 / (15 mH) = 66.66 H^{-1}



Note: The initial condition of the second integrator can be set to 0.05, but it should be connected to -1, because the integrator has an inverting output. Not +1 as in this drawing. Amplitude scaling will be described later.

Problem: I'm not strictly following my own rules here. The units of the signals are missing. This should be corrected.

Rule 3:

Change the time scale factor β to the desired value. To do this, multiply the time scale factors β of <u>all</u> integrators by a suitable factor X and multiply the time constants TC of <u>all</u> integrators by the same factor X.

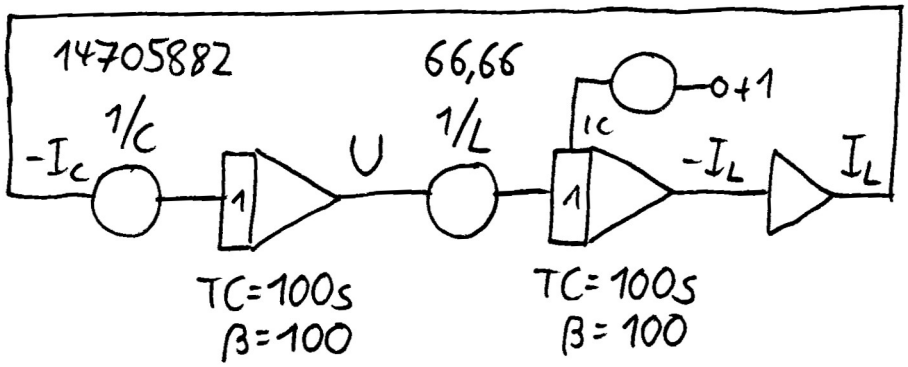
If the circuit shall run in realtime (β=1), then this step is unnecessary.

Write the new time scale factors β and time constants TC below each integrator.

All integrators in the circuit must have the same time scale factor β.

If the circuit contains delay lines, these must be changed as well.

In this example, write "TC = 100s, β = 100" below each integrator.



Now the time scale factor β is correct.

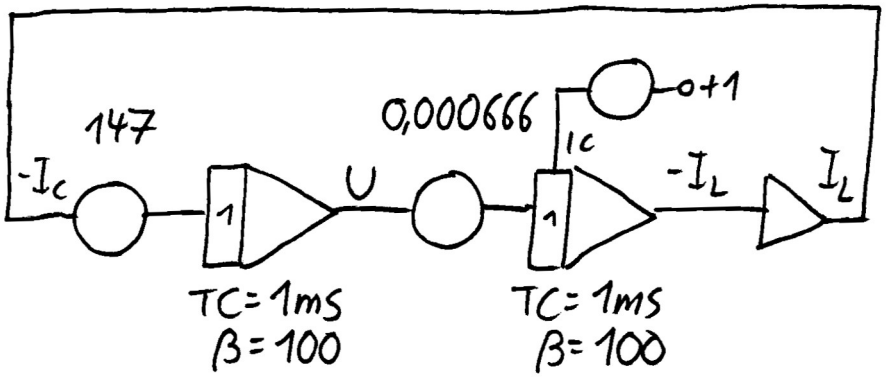
Rule 4:

The time constant TC of an integrator can be multiplied by a factor X, if all inputs of this integrator are multiplied by the same factor X. This operation doesn't change the time scale factor β and the initial condition of the integrator. Each integrator can have its own factor X.

Choose X so that the time constant of the integrator becomes either 1 ms or 100 ms, because these time costants are available in THAT (1 ms in normal mode, and 100 ms in SLOW mode).

The integrators may have different time constants TC, but they must all share the same time scale factor β.
With other words: The time constant TC is a property of each integrator, however the time scale factor β is a property of the whole circuit.

In this example the factor X is 0.00001, so that the time constants of both integrators is now 1ms, which is the default time constant of THAT's integrators.



Now the time constants TC of the integrators are correct, and the time scaling factor β is also correct. But there are still two problems: The first coefficient is too large, and the second coefficient is too small.

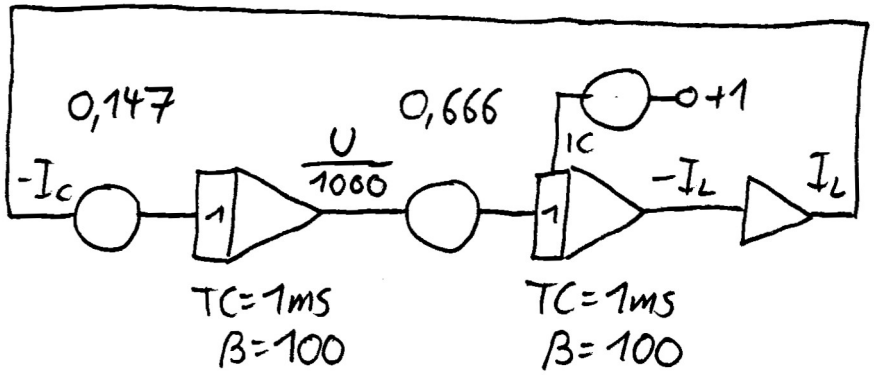
Rule 5:

If all input signals of an integrator (or summer) are multiplied by a factor X, then the output of the integrator (or summer) does also become larger by factor X. This introduces an error that must be compensated by inserting a 1/X coefficient at the output of the integrator (or summer).

If the integrator's initial condition (IC) is different from 0, then it must also be multiplied by factor X.

This operation doesn't change the integrator's time constant TC and time scale factor β.

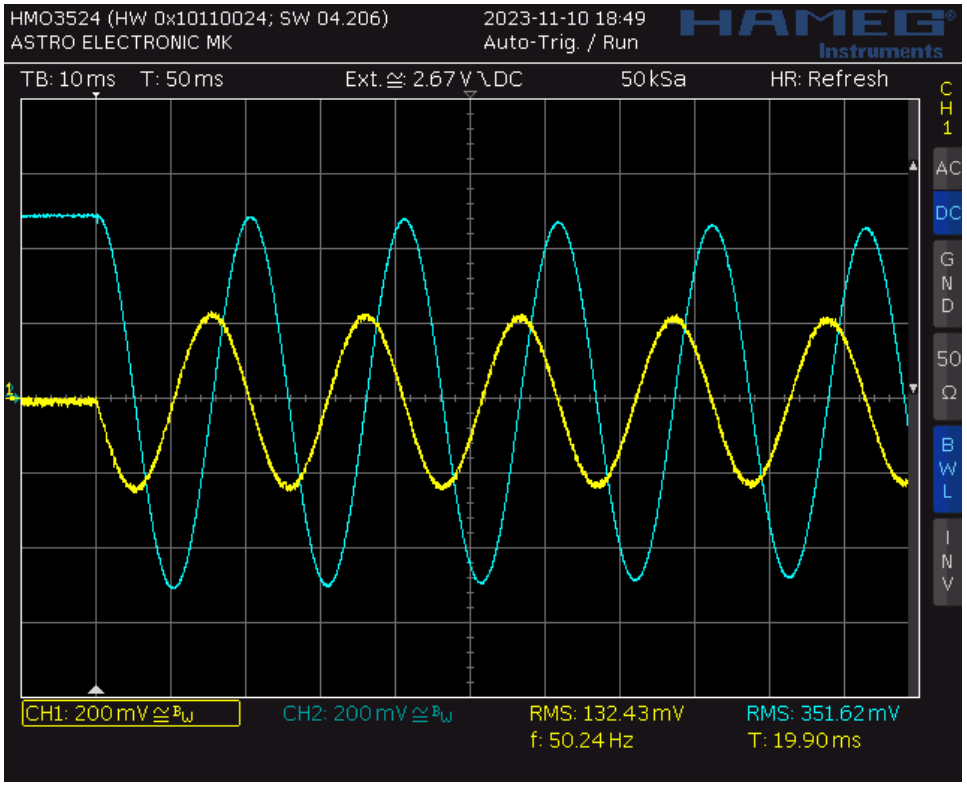
In this example, multiply the value of the first coefficient by 0.001, so that the input signal of the first integrator becomes smaller, and the output is now U / 1000.
Luckily the first integrator doesn't have a non-zero initial condition, so we don't have to change it.
We compensate the error by multiplying the second coefficient by 1000, so that the input of the second integrator gets the same signal as before.
Now we can let this circuit run and measure the resonance frequency on the oscilloscope. The frequency is about 50 Hz and the period is about 20 ms.
To get the real-world frequency, we must multiply the measured frequency by the time scale factor β = 100, which gives 5 kHz.
To get the real-world period (or time), we must divide the measured period (or time) by the time scale factor β = 100, which gives 0.2 ms.



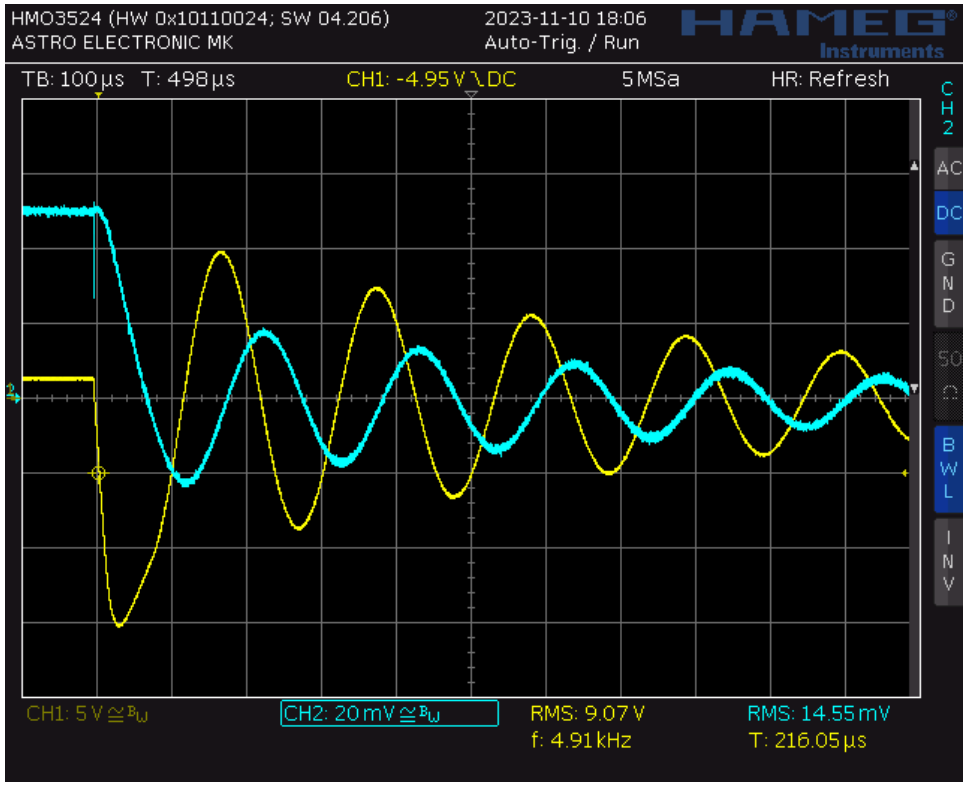
With the circuit as above (after applying rule 5) and with the initial condition set to 0.05 (Ampere) the output of the first integrator is the voltage in Millivolts (U / 1000), and the output of the second integrator is the current in Ampere.

Amplitude scaling is very simple in this example, because all amplitudes do only depend on one input. That's the initial condition of the second integrator. But 0.05 is a little bit too small for accurate results. It's better to change the amplitude scaling. For example if we change the scaling factor of the input from 1 A/mu to 0.1 A/mu, then the voltage output will change accordingly from 1 mV/mu to 0.1 mV/mu (mu = machine unit).

This is the output of the analog computer (with initial condition = -0.05).
Yellow is the output of the first integrator, which should be the voltage in Millivolts, about 1.1 div peak, which must be multiplied by 1000, which gives 1100 div peak.
Cyan is the output of the second integrator after the inverter, which should be the current in Ampere, about 2.4 div peak.
The U / I ratio is about 1100 / 2.4 = 458



This is the measurement of the real-world LC circuit. Obviously it has more damping, but that was to be expected.
Yellow is the voltage with 5 V/div.
Cyan is the current, which was measured with a 1 Ohm resistor in series with the inductor, so that 20 mV/div is equivalent to 20 mA/div.
The U / I ratio is about 450-500



7.2 Gravity on other Planets

In this example we want to simulate how fast a stone is falling from height 1m to the surface of a planet. That depends on the gravitational acceleration at the surface of the planet.

These are the formulas for the analytical solution:

Height as a function of acceleration and time: $h = a / 2 \cdot t^2$ (height is 0 at the start position, and then becomes negative as the stone is falling down)

Time for falling from height h to the ground: $t = \text{sqrt}(2 \cdot h / a)$

Velicity of falling stone after 1s: $v = a \cdot 1s$

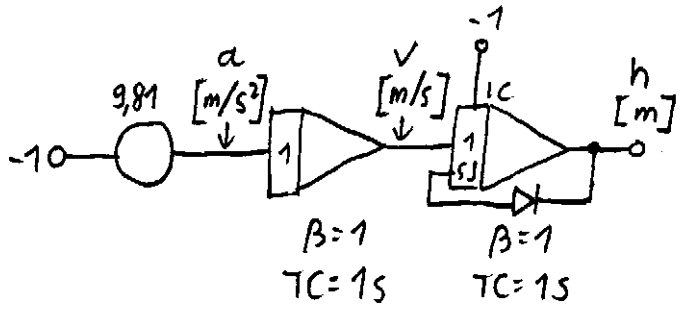
Impact velocity of stone at the ground, after falling from height h: $v_{max} = \text{sqrt}(2 \cdot a \cdot h)$

Planet	Gravitational acceleration at planet surface a [m/s²]	Time for stone falling from 1m height [s]	Impact velocity after falling from 1m height [m/s]
Earth	-9.81	0.45	-4.43
Moon	-1.62	1.11	-1.80
Mars	-3.71	0.73	-2.72
Comet 67P/Churyumov–Gerasimenko https://en.wikipedia.org/wiki/67P/Churyumov%E2%80%93Gerasimenko Escape velocity is about 1 m/s	about -0.001	44.7	-0.045

See also: How far can you throw a ball on another world? https://www.facebook.com/UnversalSci/videos/780443294144501/

See also: https://permondes.de/gitweb/Analog_Engine.git/blob/4124db17783ef95a2a0020b4a7624b41c0f1b0e5:/Potentials and Motion.odt

Rules	Example
Rule 1: As the first step, describe the problem with SI units (m, s, N, kg, V, A ...). Write down the equations. If already known, write down the desired time scale factor β. β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems. β = 1 means the simulation is running in realtime. β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions or electronic circuits. If the time scale factor isn't yet known, begin with a reasonable assumption.	At t = 0 the stone is at height h = 1m and begins to fall. The simulation ends when it reaches the ground at height h = 0. The time scale factor is β = 1 because we want to simulate in realtime. Equations and given data: Height h [m] Initial condition: h = 1 m End condition: h = 0 m Velocity: v = dh / dt [m/s] Acceleration: a = dv / dt [m/s^2] a = 9.81 m/s^2

Rule 2: **Draw the schematic diagram for the analog computer, under the assumption that** • **All integrators have a time constant TC = 1s** • **All integrators have a time scale factor β = 1** **Write "TC = 1s" and "β = 1" below each integrator.** **All signals should have a physical unit attached to them, which is normally written in [ ] brackets.** **The resulting schematic diagram should be mathematically correct, but it won't run on the analog computer because it still has many problems:** • **The time scale factor may be different from the desired value.** • **The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode).** • **The coefficients may be larger than +1 or smaller than -1 or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.05 and 1. In some cases 0.01 may be acceptable as well.**	 **Note: In this (and all following) drawings a minus sign is missing. The output signal of the velocity integrator is -v.**
Rule 3: **Change the time scale factor β to the desired value. To do this, multiply the time scale factors β of <u>all</u> integrators by a suitable factor X and multiply the time constants TC of <u>all</u> integrators by the same factor X.** **If the circuit shall run in realtime (β=1), then this step is unnecessary.** **Write the new time scale factors β and time constants TC below each integrator.** **All integrators in the circuit must have the same time scale factor β.** **If the circuit contains delay lines, these must be changed as well.**	**Nothing must be done in this example, because the simulation shall run in realtime and the time scale factor β is already correct.**

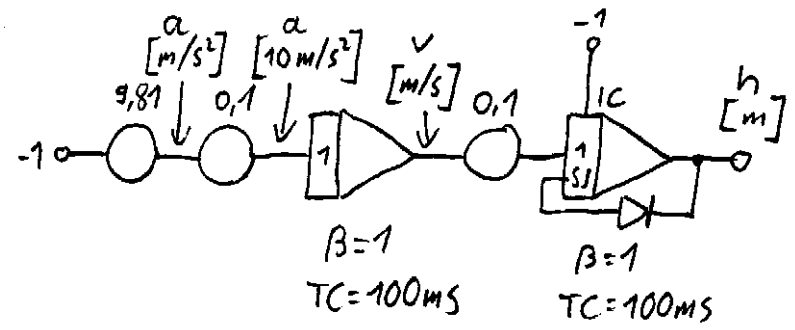
Rule 4:

The time constant TC of an integrator can be multiplied by a factor X, if all inputs of this integrator are multiplied by the same factor X. This operation doesn't change the time scale factor β and the initial condition of the integrator. Each integrator can have its own factor X.

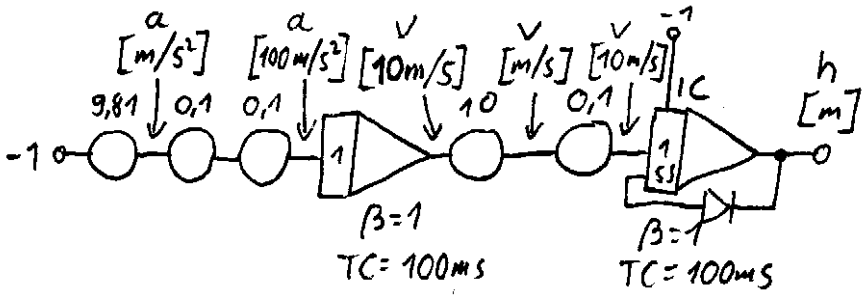
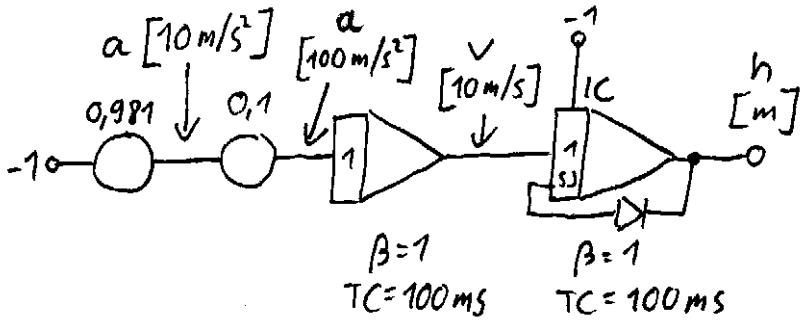
Choose X so that the time constant of the integrator becomes either 1 ms or 100 ms, because these time costants are available in THAT (1 ms in normal mode, and 100 ms in SLOW mode).

The integrators may have different time constants TC, but they must all share the same time scale factor β.
With other words: The time constant TC is a property of each integrator, however the time scale factor β is a property of the whole circuit.

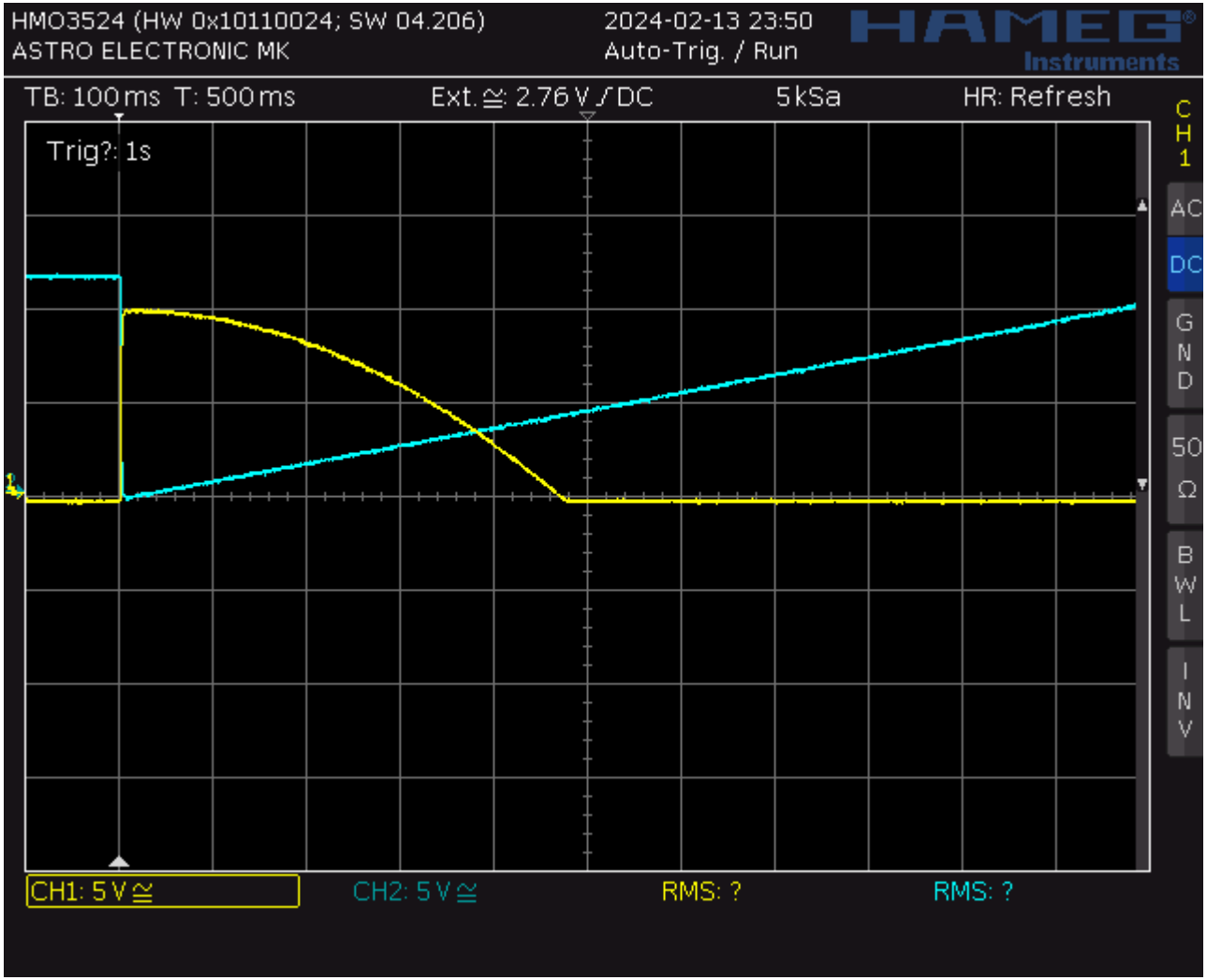
In this example the factor X is 0.1, so that the time constants of both integrators becomes 100ms, which is the time constant of THAT's integrators in SLOW mode.



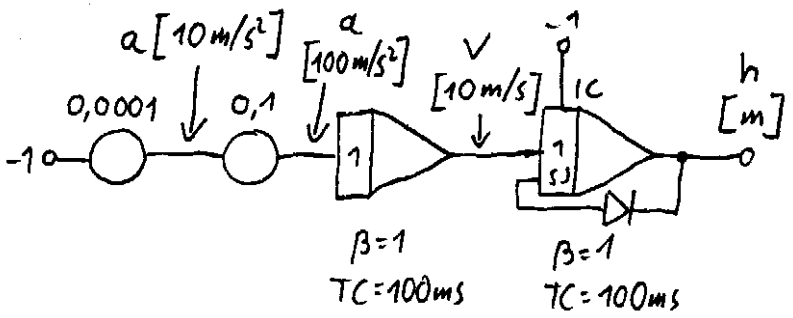
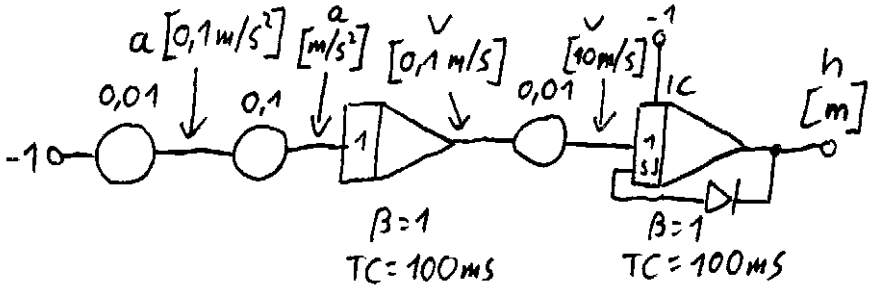
Now the time constants TC of the integrators are correct, and the time scaling factor β is also correct. But there are still two problems: The first coefficient is too large, and we know that the velocity will become 4.43 m/s, which exceeds the machine unit. The velocity integrator would overflow before the stone has reached the ground.

Rule 5: **If all input signals of an integrator (or summer) are multiplied by a factor X, then the output of the integrator (or summer) does also become larger by factor X. This introduces an error that must be compensated by inserting a 1/X coefficient at the output of the integrator (or summer).** **If the integrator's initial condition (IC) is different from 0, then it must also be multiplied by factor X.** **This operation doesn't change the integrator's time constant TC and time scale factor β.**	**Let's solve the second problem first. We make the input of the velocity integrator smaller by a factor 10, so that its output does also become smaller by a factor 10. To compensate the error, we multiply the output signal by a factor 10.** 
Rule 6: **Multiple coefficients in series can be combined into a single coefficient, which is the product of them.** **If the coefficient is 1, it can be removed.**	**We combine the 9.81 and 0.1 coefficients to a single 0.981 coefficient, which has now a different unit [10 m/s^2]. The other 0.1 coefficient remains as it is.** **At the output of the velocity integrator, the 10 and 0.1 coefficients compensate each other and can be removed. The unit of the velocity signal is now [10 m/s].** 

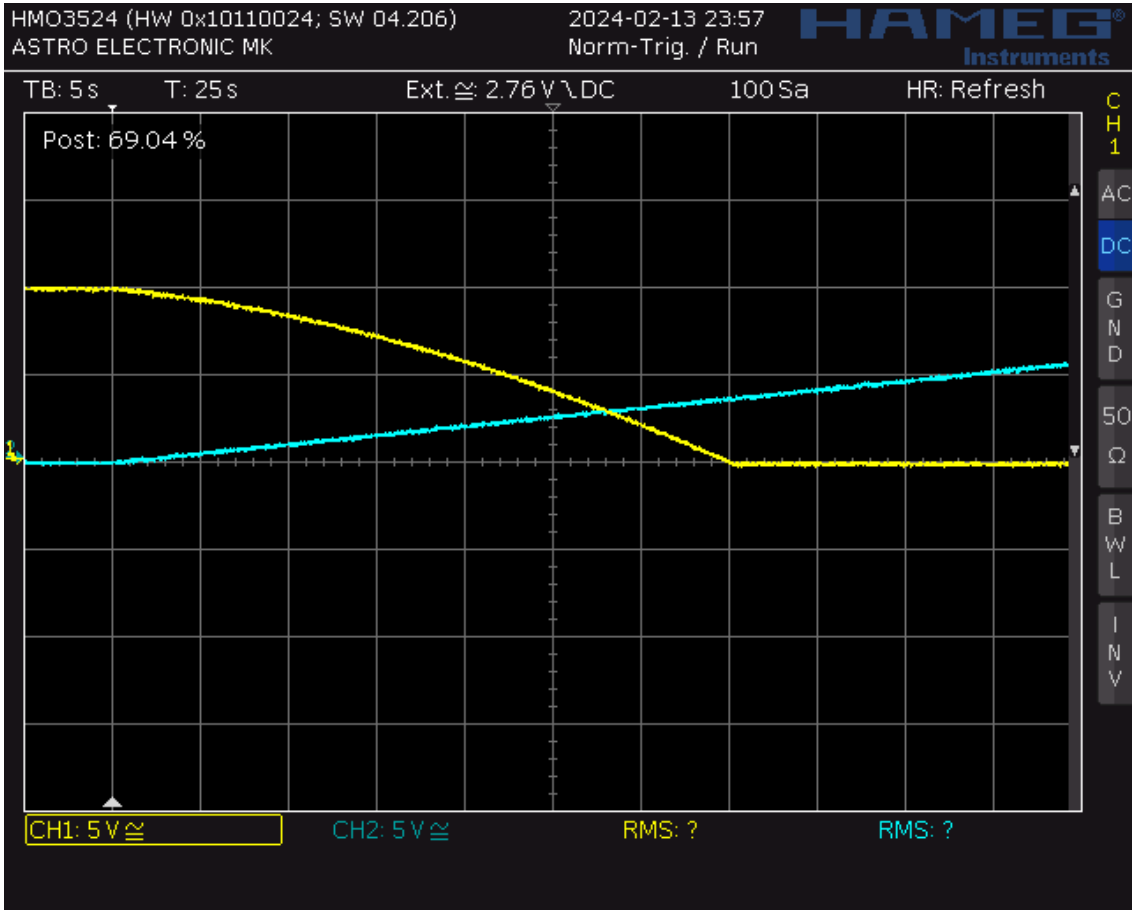
This is the result of the simulation for acceleration = 9.81 m/s^2. Yellow is the height and cyan is the inverted velocity. It takes about 0.47s for the stone to fall from 1m height. Which is in good agreement with 0.45s from the analytical solution.



Now let's try the same simulation for comet 67P/Churyumov–Gerasimenko, which has only 0.001 m/s² gravitational acceleration, as opposed to 9.81 m/s² on earth.

This is the same circuit as in the last drawing, only the gravitational acceleration was changed from 9.81 m/s² to 0.001 m/s², which is 0.0001 with unit [10 m/s²]. **Obviously the coefficient 0.0001 is much too small. What can be done?**	
Rule 8: **If a signal is sent through a coefficient, then the output signal can be interpreted as the same signal but with a different unit.** **Example:** **The input signal is a velocity with unit [m/s].** **The coefficient is 0.1.** **The output signal is the same velocity with unit [10 m/s], because 1 / 0.1 = 10.** **Important things to remember:** **If the coefficient is smaller than 1, then the output unit becomes larger.** **If the coefficient is larger than 1, then the output unit becomes smaller.**	**Increase the first coefficient by a factor 100, to bring it to an acceptable value. By doing this, the unit of the gravitational acceleration changes from 10 m/s² to 0.1 m/s².** **The input of the velocity integrator is now too large by a factor 100, and the output is too large by this factor as well. To compensate this error, we insert a 0.01 coefficient at the output of the velocity integrator.** 
One more check is required:	**Could the velocity integrator's output overflow?** **From the analytical solution we know that the expected impact velocity is -0.045 m/s, that's -0.45 machine units because the unit is 0.1 m/s.** **The integrator won't overflow.**

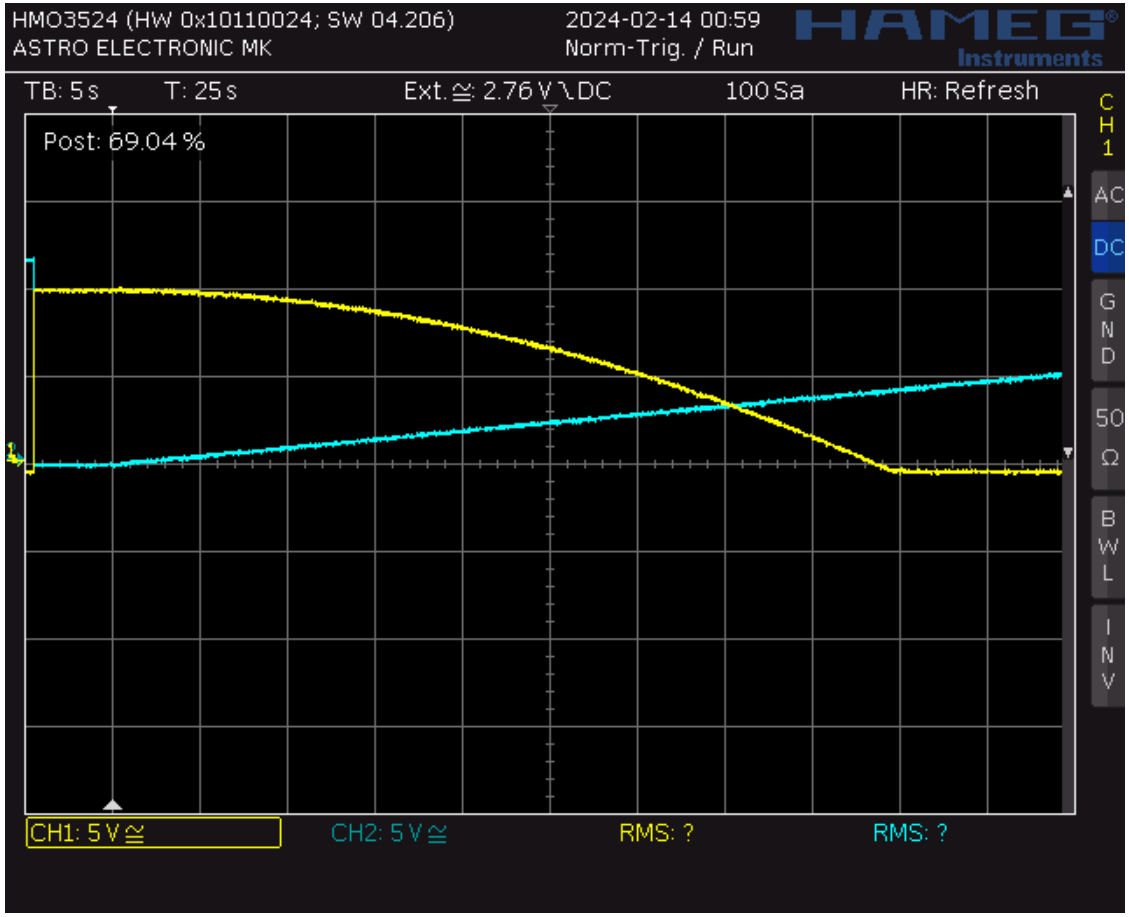
This is the output of the simulation for comet 67P/Churyumov–Gerasimenko. It takes about 35s for the stone to fall from 1m height.
This is not in good agreement with 44.7s from the analytical solution. What's wrong?



I figured out that the reverse current of the diode is the problem. The diodes in THAT are BAS170W Schottky diodes, which have a maximum reverse current of 100 nA.

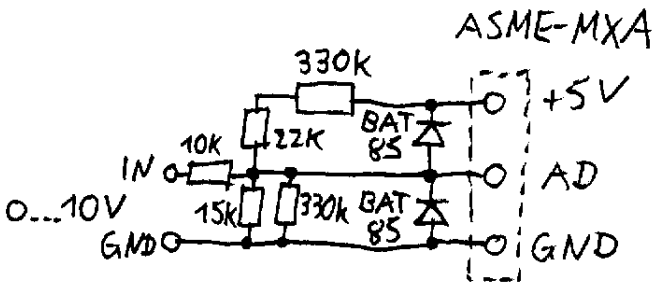
In this case it's better to use a 1N4148 diode, which has a maximum reverse current of 25 nA, or even better a BAS34 diode which has an ultra-small 1 nA reverse current.

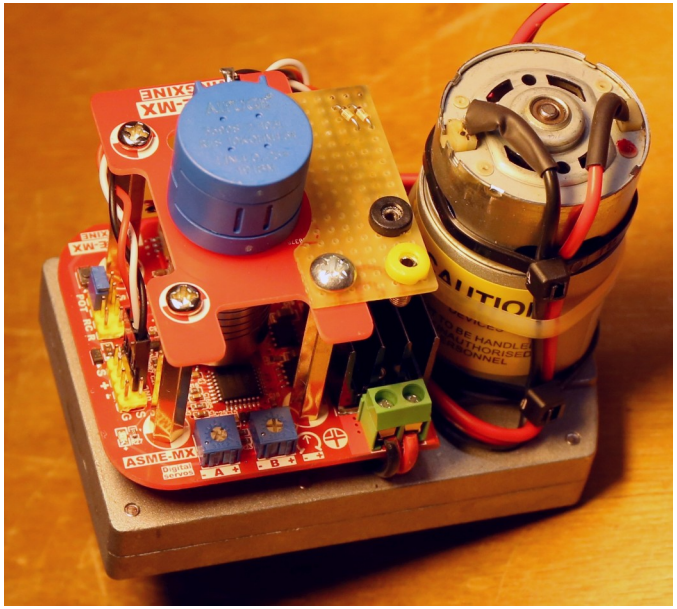
This is the output with a BAS34 diode. 47s is in good agreement with 44.7s from the analytical solution:

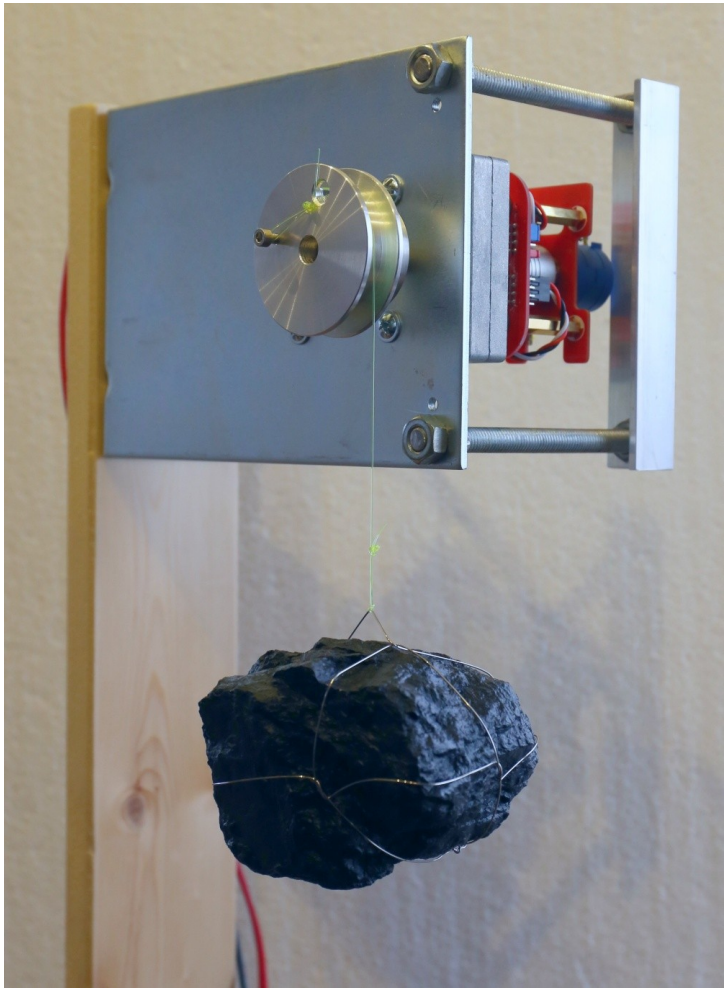


For educational purpose I want to demonstrate how slow the stone is falling. The stone hangs at the end of a string which is wound on a pulley, rotated by a servo motor.

I'm using the ASME-MXA servo motor, which has a maximum speed of 0.72s per revolution at 24V. These high-power servos have a pulse width input (like other servos), but they also have a 0 to 5V analog voltage input. This input can easily be converted to a 0 to 10V input with the circuit shown below. Then you can control the angle of the output shaft with a signal from THAT. The maximum angle is about 9.5 revolutions or 3420°, but can also be set to a smaller range with a potentiometer. If the travel range is 1m for 9 revolutions, the diameter of the pulley must be 35.4mm. The maximum speed is about 6.5 s for 1 m movement.



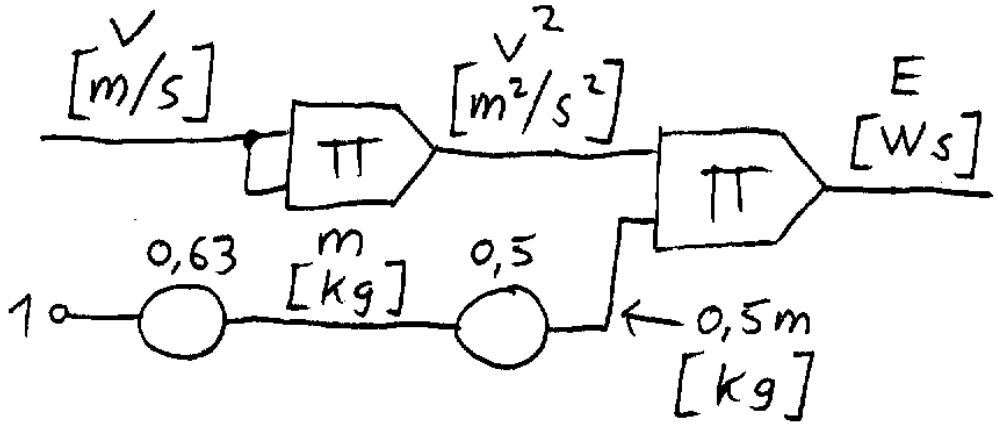


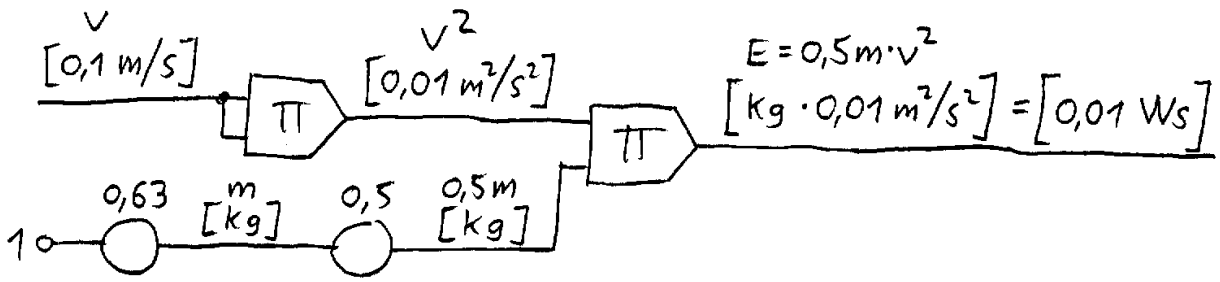
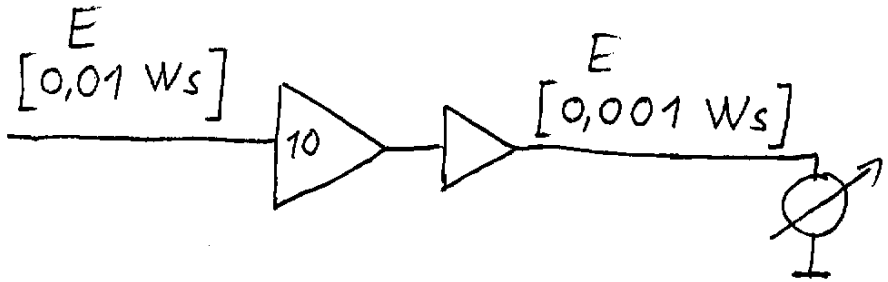


What happens if the surface of the comet is elastic?

The elastic surface acts like a spring. The kinetic energy of the stone ($0.5 \cdot m \cdot v^2$) is stored as spring energy ($0.5 \cdot k \cdot h^2$) in the compressed spring, and then released in opposite direction as kinetic energy to the stone. In absence of energy loss due to damping, the stone would bounce off the surface and return to its initial position at 1 m height.

Let's first calculate the kinetic energy with the analog computer.

Rule 1: **As the first step, describe the problem with SI units (m, s, N, kg, V, A ...). Write down the equations.**	**$E = 0.5 \cdot m \cdot v^2$** **with E = Energy in Ws (or Nm)** **m = Mass in kg, here m = 0.63 kg** **v = Velocity in m/s, from the analytical solution we already know that the impact velocity is -0.045 m/s.** **The inpact energy should be $E = 0.5 \cdot 0.63\ kg \cdot (-0.045\ m/2)^2 = 0.000638\ Ws$**
Rule 2: **Draw the schematic diagram for the analog computer.**	 **This circuit is mathematically correct, but it has one problem:** **The velocity from the previous circuit is given with a different unit, either [0.1 m/s] or [10 m/s].**

Scaling Rule 11: **If two problem variables are multiplied, the output unit becomes the product of the units of the two input factors.**	**In this case it's better to use the velocity signal with unit [0.1 m/s] because it has the larger machine variable value.**  **Because we already know from the analytical solution that the impact energiy is 0.000638 Ws, the output value would be: 0.0638 · 0.01 Ws = 0.000638 Ws.**
Scaling Rule 8: **If the signal is multiplied by factor 10, the unit of the output signal is divided by 10.**	**To make the output signal better readable on an analog voltmeter, we can multiply it by a factor 10:**  **The inverter is required because the summer is inverting as well.**
Scaling Rule 9: **If you connect a signal to an analog 10V meter, the unit in [ ] brackets becomes the full scale value of the voltmeter.**	**The full scale value of the analog voltmeter is 1 machine unit = 0.001 Ws.**

An elastic spring can be described by Hooke's law: https://en.wikipedia.org/wiki/Hooke%27s_law

$k = F / h$

with k = stiffness of the spring [N/m] See also: https://en.wikipedia.org/wiki/Stiffness

F = force at the spring [N]

h = displacement of the spring [m]

The energy stored in the spring is:

$E = 0.5 \cdot k \cdot h^2$

Let's assume the stone with mass 0.63 kg does penetrate 0.03 m deep into the soft surface of the comet, until all of its kinetic energy is converted to spring energy.

Then we can calculate the stiffness of the spring:

$k = E / (0.5 \cdot h^2) = 0.000638 \text{ Ws} / (0.5 \cdot 0.03^2 \text{ m}^2) = 1.418 \text{ N/m}$

How can the elastic surface be modeled in the analog computer?

If the stone touches the comet's surface, the stone's height becomes negative and the spring produces an additional acceleration a_s acting on the stone:

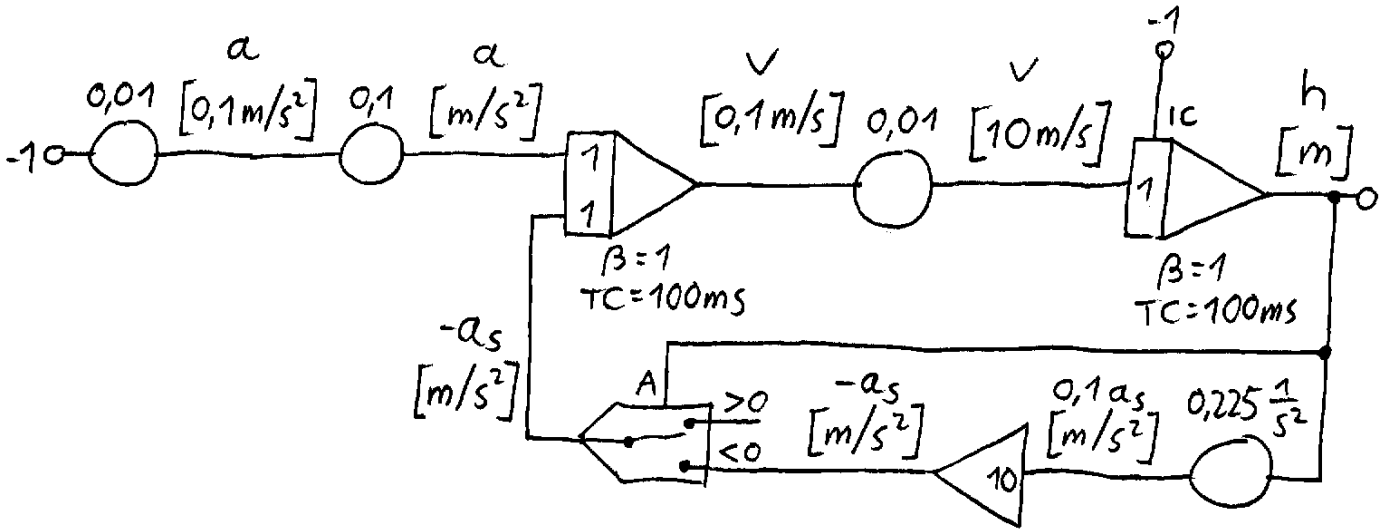
$F = m \cdot a_s$ or $a_s = F / m$, where $F = k \cdot h$

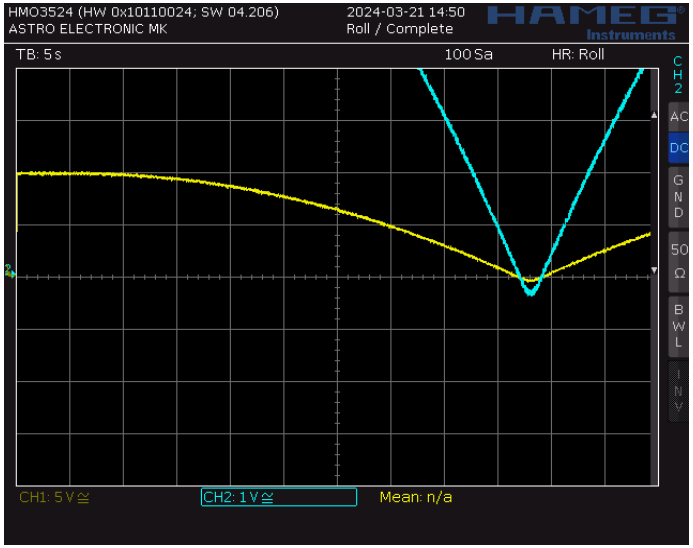
$a_s = h \cdot k / m$

k / m is a constant: $k / m = 1.418 \text{ N/m} / 0.63 \text{ kg} = 2.25 \cdot 1/\text{s}^2$

Acceleration from the spring, only active if the height is negative: $a_s = h \cdot 2.25 \cdot 1/\text{s}^2$

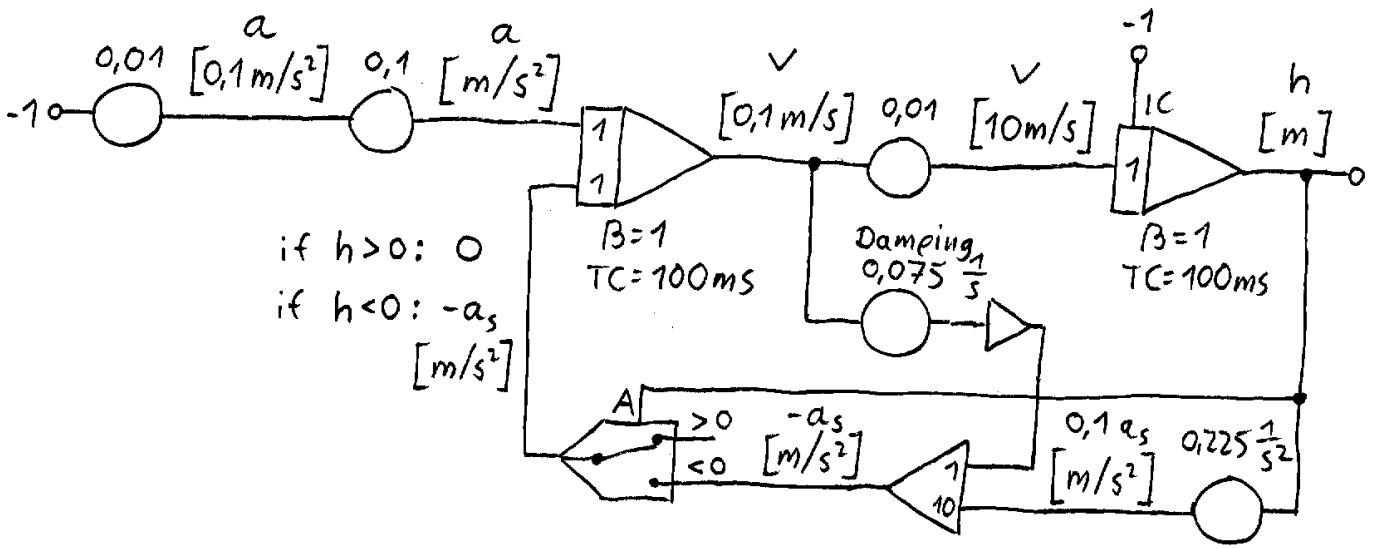
This is the circuit with bouncing on the surface, without any energy losses. The diode at the height integrator is no longer required:

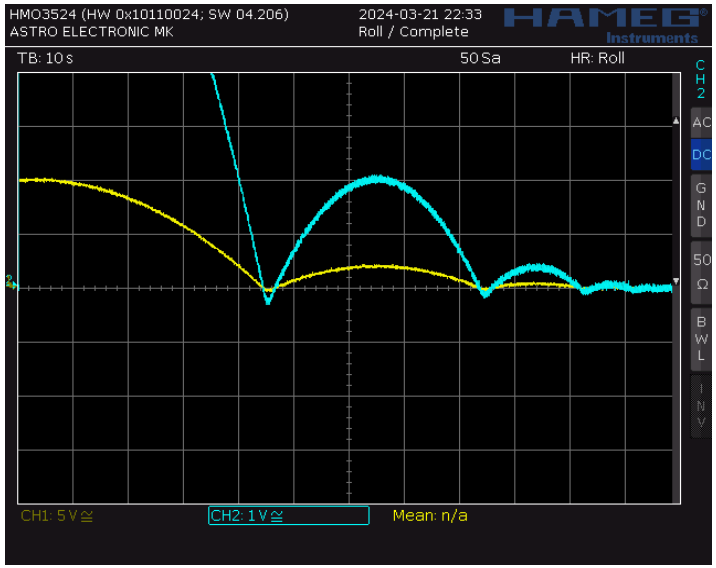




This is the output. Yellow is the height of the stone with 0.5 m/div, and cyan is the same signal enlarged to 0.1 m/div. As you can see, the spring is compressed exactly 0.03 m.

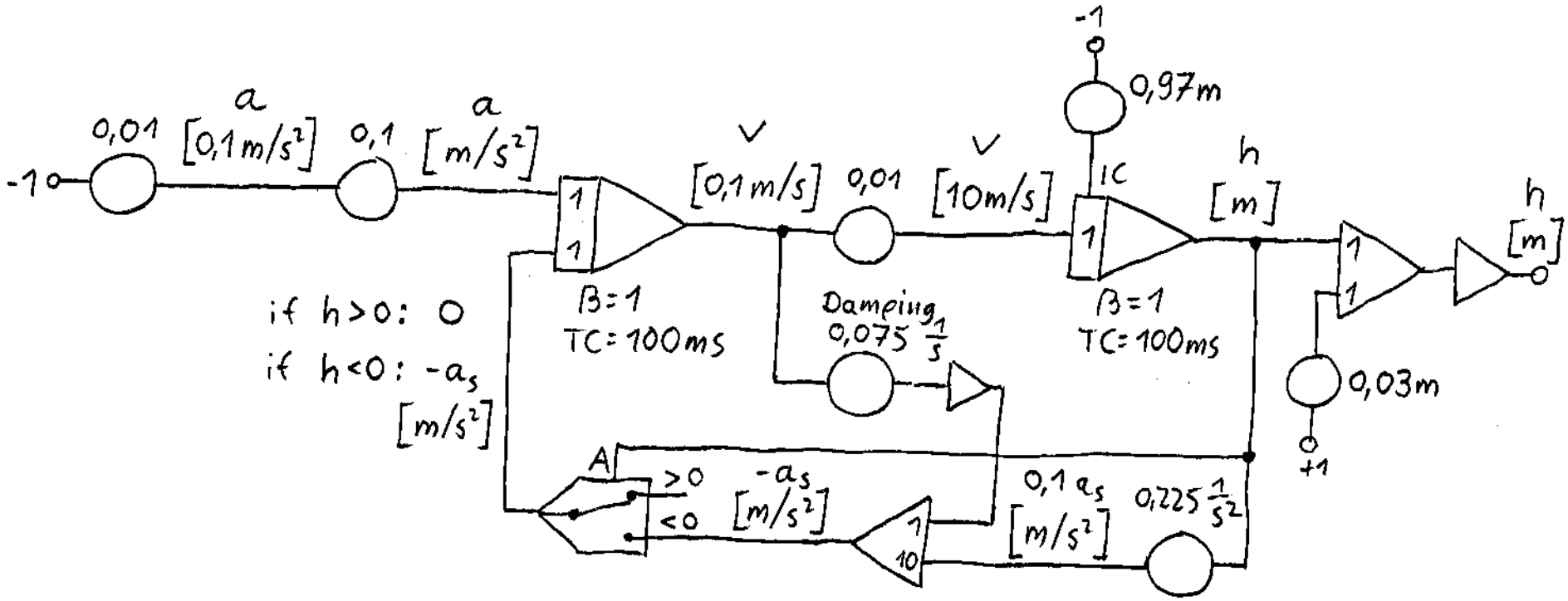
Now let's add damping when the stone penetrates into the comet's surface:





This is the output. Yellow is the height of the stone with 0.5 m/div, and cyan is the same signal enlarged to 0.1 m/div.

The height signal in the previous circuit goes from the initial position at h = 1 m to the mininum position at h = -0.03 m. That means the amplitude is a little too big for the servo motor, which accepts only positive signals from 0 to 1 machine unit. This problem is solved in the below circuit by changing the initial condition to 0.97 m and adding 0.03 m to the output signal.



See also chapter 9.8 in the first steps booklet: https://the-analog-thing.org/THAT_First_Steps.pdf

7.3 Gravitational Wave Simulation

This chapter is about the simulation of gravitational waves from the last 10 seconds before a black hole or neutron star merge. The signals are typically in the 1 Hz to 100 Hz range and you can hear them with a subwoofer.

Frank Ohme described in his article "Schwarze Löcher und andere Mysterien" some simple formulas that describe gravitational waves:

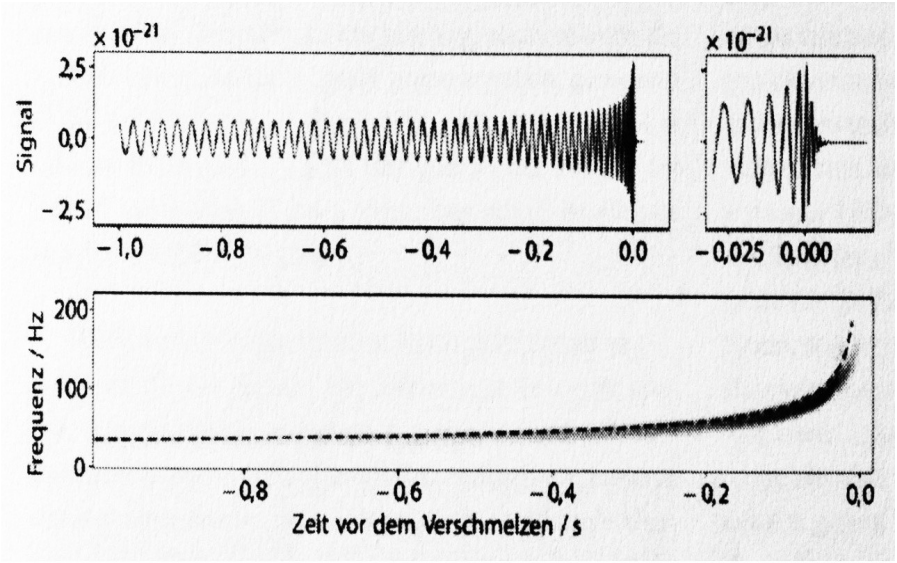
Source: https://onlinelibrary.wiley.com/doi/epdf/10.1002/piuz.202101618

Die Umlaufsfrequenz nimmt wie oben beschrieben immer schneller zu und führt zu einem für das System charakteristischen Verlauf der Gravitationswelle: dem „Chirp". In grober Näherung kann dieser beschrieben werden als

$$f(t) = \frac{1}{8\pi} \cdot \left(\frac{c^3}{GM_c}\right)^{5/8} \cdot \left(\frac{5}{\delta t}\right)^{3/8},$$

wobei f die Frequenz des Gravitationswellensignals ist, c ist die Lichtgeschwindigkeit, G die Gravitationskonstante, δt die Zeit bis zur Verschmelzung. M_c ist die sogenannte Chirpmasse: eine für das System charakteristische Masse, die aus den Massen m_1 und m_2 der beiden umeinander kreisenden Objekte errechnet werden kann:

$$M_c = \frac{(m_1 \cdot m_2)^{3/5}}{(m_1 + m_2)^{1/5}}.$$



Quelle: https://onlinelibrary.wiley.com/doi/epdf/10.1002/piuz.202101618

The drawing at the ight is a theoretical prediction of the last second of the gravitational wave signal of two collapsing black holes with 10 and 15 solar masses.

This is the formula for frequency over time (the frequency is exact, except for the last 0.1 seconds before merger):

$$F(t) = 1/(8\pi) \cdot (c^3/(GM_C))^{5/8} \cdot (5/\delta t)^{3/8}$$

with $c = 3 \cdot 10^8$ m/s — Light speed
$G = 6.67 \cdot 10^{-11}$ $m^3/(kg\ s^2)$ — Gravitational constant
$M_C = (m_1 \cdot m_2)^{3/5} / (m_1 + m_2)^{1/5}$ — Chirp mass, has the same unit as m_1 and m_2.
m_1, m_2 — Mass of both objects in kg

It's useful to express the masses in solar masses:
$m_\odot = 1{,}99 \cdot 10^{30}$ kg

The formula can be simplified to:

$F(t) = 151.2\ s^{-5/8} \cdot M_C^{-5/8} \cdot (\delta t)^{-3/8}$ where M_C must be in solar masses

For $m_1 = 10\ m_\odot$ **and** $m_2 = 15\ m_\odot$ **we get:**
$M_C = 10.6\ m_\odot$
$F(t) = 34.5\ s^{-5/8} \cdot (\delta t)^{-3/8}$ This is in agreement with drawing 2 (Abb. 2) in the article from Frank Ohme.

There is a lower limit for the mass of stellar black holes, at about 2 solar masses.

For $m_1 = m_2 = 2\ m_\odot$ **we get:**
$M_C = 1.74\ m_\odot$
$F(t) = 106.9\ s^{-5/8} \cdot (\delta t)^{-3/8}$

The chirp mass M_C has the same unit as m_1 and m_2.

$$M_C = (m_1 \cdot m_2)^{3/5} / (m_1 + m_2)^{1/5}$$

Table for the chirp mass, depending on masses m_1 and m_2:

	m_2											
m_1	2 $m_\odot$	3 $m_\odot$	4 $m_\odot$	5 $m_\odot$	6 $m_\odot$	8 $m_\odot$	10 $m_\odot$	12 $m_\odot$	15 $m_\odot$	20 $m_\odot$	30 $m_\odot$	50 $m_\odot$
2 $m_\odot$	1.74 $m_\odot$											
3 $m_\odot$	2.12 $m_\odot$	2.61 $m_\odot$										
4 $m_\odot$	2.42 $m_\odot$	3.01 $m_\odot$	3.48 $m_\odot$									
5 $m_\odot$	2.70 $m_\odot$	3.35 $m_\odot$	3.89 $m_\odot$	4.35 $m_\odot$								
6 $m_\odot$	2.93 $m_\odot$	3.65 $m_\odot$	4.25 $m_\odot$	4.76 $m_\odot$	5.22 $m_\odot$							
8 $m_\odot$	3.33 $m_\odot$	4.17 $m_\odot$	4.87 $m_\odot$	5.48 $m_\odot$	6.02 $m_\odot$	6.96 $m_\odot$						
10 $m_\odot$	3.67 $m_\odot$	4.61 $m_\odot$	5.40 $m_\odot$	6.08 $m_\odot$	6.79 $m_\odot$	7.78 $m_\odot$	8.71 $m_\odot$					
12 $m_\odot$	3.97 $m_\odot$	5.00 $m_\odot$	5.86 $m_\odot$	6.62 $m_\odot$	7.30 $m_\odot$	8.49 $m_\odot$	9.53 $m_\odot$	10.4 $m_\odot$				
15 $m_\odot$	4.37 $m_\odot$	5.51 $m_\odot$	6.47 $m_\odot$	7.33 $m_\odot$	8.09 $m_\odot$	9.44 $m_\odot$	10.6 $m_\odot$	11.7 $m_\odot$	13.1 $m_\odot$			
20 $m_\odot$	4.93 $m_\odot$	6.23 $m_\odot$	7.34 $m_\odot$	8.33 $m_\odot$	9.22 $m_\odot$	10.8 $m_\odot$	12.2 $m_\odot$	13.4 $m_\odot$	15.0 $m_\odot$	17.4 $m_\odot$		
30 $m_\odot$	5.83 $m_\odot$	7.39 $m_\odot$	8.73 $m_\odot$	9.93 $m_\odot$	11.0 $m_\odot$	12.9 $m_\odot$	14.6 $m_\odot$	16.2 $m_\odot$	18.2 $m_\odot$	21.2 $m_\odot$	26.1 $m_\odot$	
50 $m_\odot$	7.19 $m_\odot$	9.14 $m_\odot$	10.8 $m_\odot$	12.3 $m_\odot$	13.7 $m_\odot$	16.2 $m_\odot$	18.3 $m_\odot$	20.3 $m_\odot$	23.0 $m_\odot$	27.0 $m_\odot$	33.5 $m_\odot$	43.5 $m_\odot$

Table for the frequency, which is produced one second before the black holes with masses m_1 und m_2 merge:

	m_2											
m_1	2 $m_\odot$	3 $m_\odot$	4 $m_\odot$	5 $m_\odot$	6 $m_\odot$	8 $m_\odot$	10 $m_\odot$	12 $m_\odot$	15 $m_\odot$	20 $m_\odot$	30 $m_\odot$	50 $m_\odot$
2 $m_\odot$	107 Hz											
3 $m_\odot$	94.4 Hz	83.0 Hz										
4 $m_\odot$	86.7 Hz	75.9 Hz	69.3 Hz									
5 $m_\odot$	81.3 Hz	71.0 Hz	64.7 Hz	60.3 Hz								
6 $m_\odot$	77.2 Hz	67.3 Hz	61.2 Hz	57.0 Hz	53.8 Hz							
8 $m_\odot$	71.3 Hz	62.0 Hz	56.2 Hz	52.2 Hz	49.2 Hz	45.0 Hz						
10 $m_\odot$	67.1 Hz	58.2 Hz	52.7 Hz	48.9 Hz	46.1 Hz	42.0 Hz	39.1 Hz					
12 $m_\odot$	63.9 Hz	55.3 Hz	50.1 Hz	46.4 Hz	43.6 Hz	39.7 Hz	37.0 Hz	34.9 Hz				
15 $m_\odot$	60.2 Hz	52.1 Hz	47.1 Hz	43.6 Hz	40.9 Hz	37.2 Hz	34.5 Hz	32.6 Hz	30.3 Hz			
20 $m_\odot$	55.8 Hz	48.2 Hz	43.5 Hz	40.2 Hz	37.7 Hz	34.2 Hz	31.7 Hz	29.9 Hz	27.8 Hz	25.2 Hz		
30 $m_\odot$	50.2 Hz	43.3 Hz	39.0 Hz	36.0 Hz	33.8 Hz	30.5 Hz	28.2 Hz	26.5 Hz	24.6 Hz	22.4 Hz	19.7 Hz	
50 $m_\odot$	44.1 Hz	37.9 Hz	34.1 Hz	31.5 Hz	29.5 Hz	26.6 Hz	24.5 Hz	23.0 Hz	21.3 Hz	19.3 Hz	16.8 Hz	14.3 Hz

Simplified formula for frequency over time:

$F(t) = F_1 \cdot (\delta t)^{-3/8}$ where F_1 is a constant, and δt is the time until merger.

The value of F_1 is the frequency which is produced one second before the black holes merge (because $1^{-3/8} = 1$).

It's useful to calculate some values in advance:

For $m_1 = 10\ m_\odot$ und $m_2 = 15\ m_\odot$:

Time δt until merger	10 s	5 s	2 s	1 s	0.5 s	0.2 s	0.1 s
Frequency F(t)	14.5 Hz	18.9 Hz	26.6 Hz	34.5 Hz	44.7 Hz	63.1 Hz	81.8 Hz

For $m_1 = 2\ m_\odot$ und $m_2 = 2\ m_\odot$:

Time δt until merger	10 s	5 s	2 s	1 s	0.5 s	0.2 s	0.1 s
Frequency F(t)	45.1 Hz	58.5 Hz	82.5 Hz	107 Hz	139 Hz	196 Hz	254 Hz

How can a function with fractional exponent be realized on the analog computer?

- **$a^x = e^{(x \cdot \ln(a))}$** This approach requires and exponential function and a logarithm function. There are available on some analog computers, but not on THAT.
- **ADC --> µC --> DAC** This is possible, but it's not an alalog solution.
- **$x^{3/8} = \sqrt[8]{x^3}$** This approach works only in theory, but has problems with dynamic range. Also we would need 5 multipliers, but we have only 2 on THAT.

But there exists a better solution ...

$F(t) = F_1 \cdot (t_m - t)^{-3/8}$

with $F(t)$ = Frequency depending on time t

F_1 = Constant (the value equals the frequency 1 second before merger)

t_m = Time of merger

t = Time

How can we realize this function with fractional exponent on the analog computer?

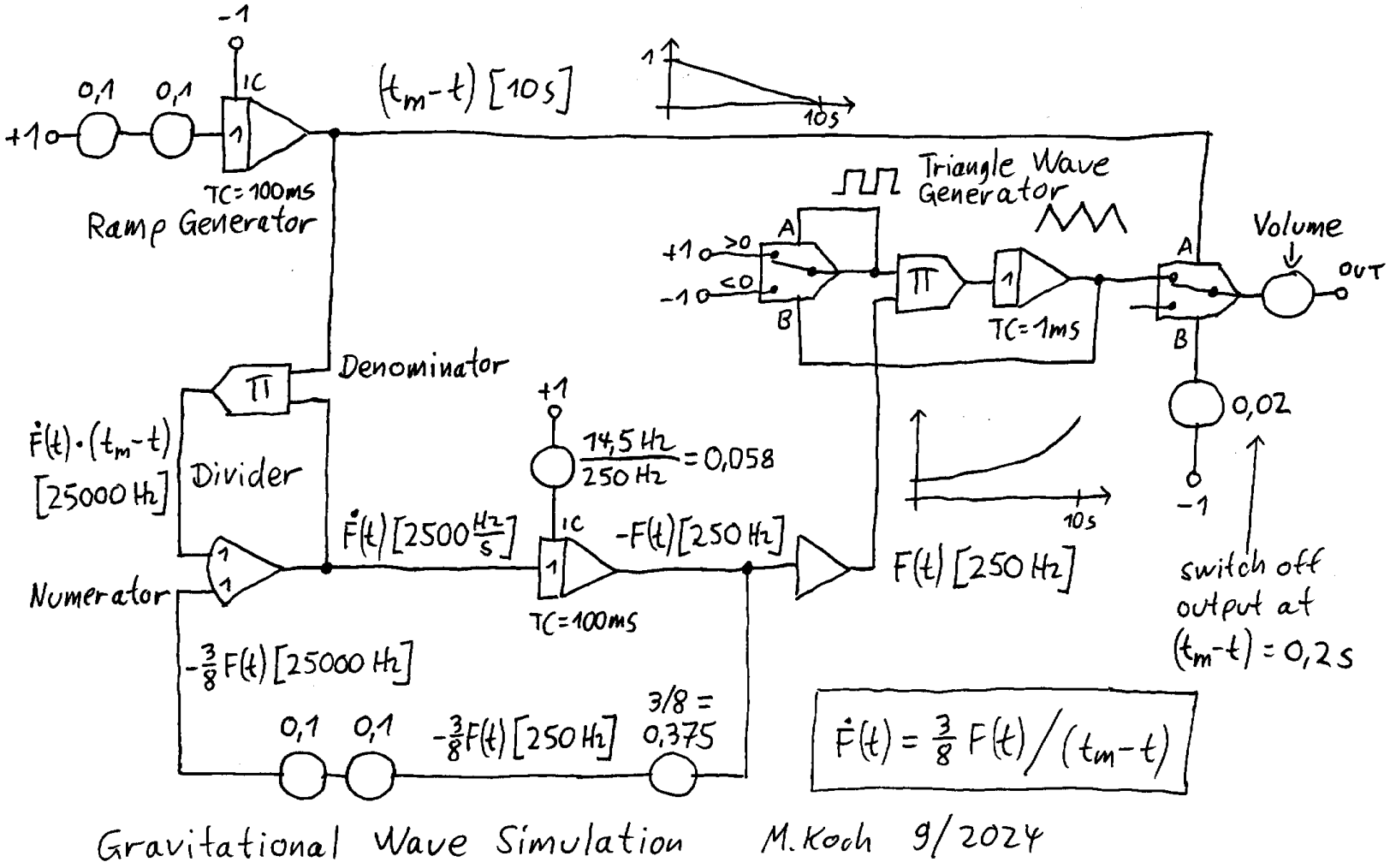
$\dot{F}(t) = F_1 \cdot -3/8 \cdot (t_m - t)^{-11/8} \cdot (-1)$ $\dot{F}(t)$ is the derivative of F(t) by the time t, don't forget the inner derivative (-1) !

$\dot{F}(t) = 3/8 \cdot F_1 \cdot (t_m - t)^{-11/8}$ Combine the signs

$\dot{F}(t) = 3/8 \cdot F_1 \cdot (t_m - t)^{-3/8} / (t_m - t)$ because $a^n = a^{n+1} / a$

$\dot{F}(t) = 3/8 \cdot F(t) / (t_m - t)$ $F_1 \cdot (t_m - t)^{-3/8}$ can be replaced by F(t)

Now we have a differential equation with $\dot{F}(t)$ and F(t), which doesn't contain a fractional exponent. This equation can be solved on the analog computer.



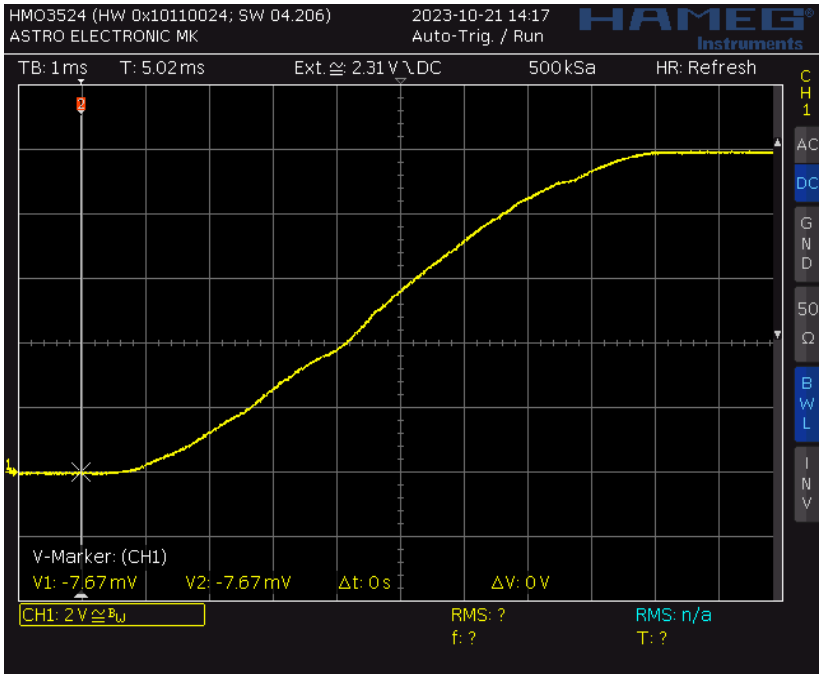
For m_1 = 10 $m_\odot$ and m_2 = 15 $m_\odot$ the initial condition is 14.5 Hz / 250 Hz = 0.058
For m_1 = 2 $m_\odot$ and m_2 = 2 $m_\odot$ the initial condition is 45.1 Hz / 250 Hz = 0.180
Hint: Due to offset errors at the input of the multiplier in the divider section, the frequency integrator might run in the wrong direction at the beginning. In this case it may help to swap the two inputs of the multiplier.

7.4 Uphill Mountainbiking

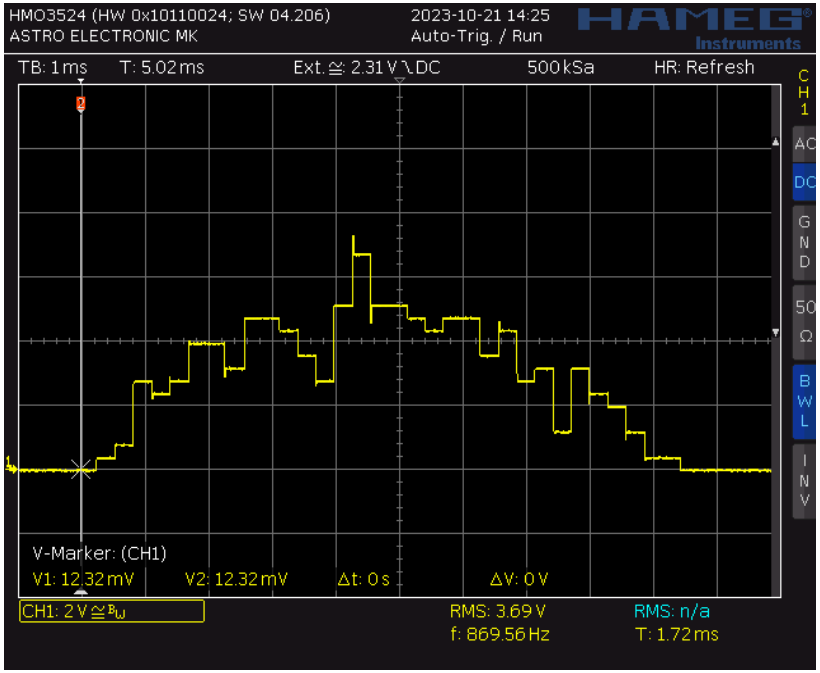
This example is a simulation for uphill mountainbiking. My favourite hill for training is the Eichelnkopf near Herzberg in the Harz Mountains in central Germany. It has 53.2 m elevation over 350 m distance, with a maximum slope of about 34%-35%. I did make three barometric measurements of the elevation profile and then calculated the average profile. This profile was programmed as a look-up-table in my Teensy LC circuit for THAT. For details see the Teensy book: http://www.astro-electronic.de/Teensy_Book.pdf

The look-up-table contains either elevation over distance, or slope over distance. In the following example I'm using the slope over distance function. The scaling factor for the distance is 1 machine unit for 350m, and the scaling factor for the slope output is 1 machine unit for 50% slope.

This is the elevation profile as a function of distance:



This is the slope as a function of distance. It's always positive because the elevation profile is monotonically increasing:



Rules	Example
Rule 1: As the first step, describe the problem with SI units (m, s, N, kg, V, A ...). Write down the equations. If already known, write down the desired time scale factor β. β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems. β = 1 means the simulation is running in realtime. β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions or electronic circuits. If the time scale factor isn't yet known, begin with a reasonable assumption.	Let's assume that the cyclist (that's me) is ascending the mountain with a constant velocity. It takes about 5 minutes to the peak. That means the velocity is 350m in about 300 seconds, which is about 1.2 m/s. Let's allow velocities up to 2 m/s in the simulation. P is the power required to move the cyclist and bicycle up the hill. $P = m \cdot g \cdot V_{VERTICAL}$ P_{PEDAL} is the power required at the pedals, it's larger because of slippage between the rear wheel and the ground, and losses in the chain and gear transmission. $P_{PEDAL} = P / Efficiency$ $V_{VERTICAL} = V_{HORIZONTAL} \cdot S$ $dx/dt = V_{HORIZONTAL}$ $dh/dt = V_{VERTICAL}$ with m = total mass of cyclist and bicycle in kg, typical value is about 95 kg g = gravitational acceleration 9.81 m/s^2 S = slope is a dimensionless factor, where 1 is equivalent to 100% slope or 45° V = velocity in m/s x = horizontal position in m h = vertical elevation in m Efficiency is a dimensionsless factor, about 0.85 Let's use a time scale factor β = 0.0001, so that the simulation runs 10000x faster that the real-world problem, in about 30 ms.

Rule 2:

Draw the schematic diagram for the analog computer, under the assumption that

- **All integrators have a time constant TC = 1s**
- **All integrators have a time scale factor β = 1**

Write "TC = 1s" and "β = 1" below each integrator.

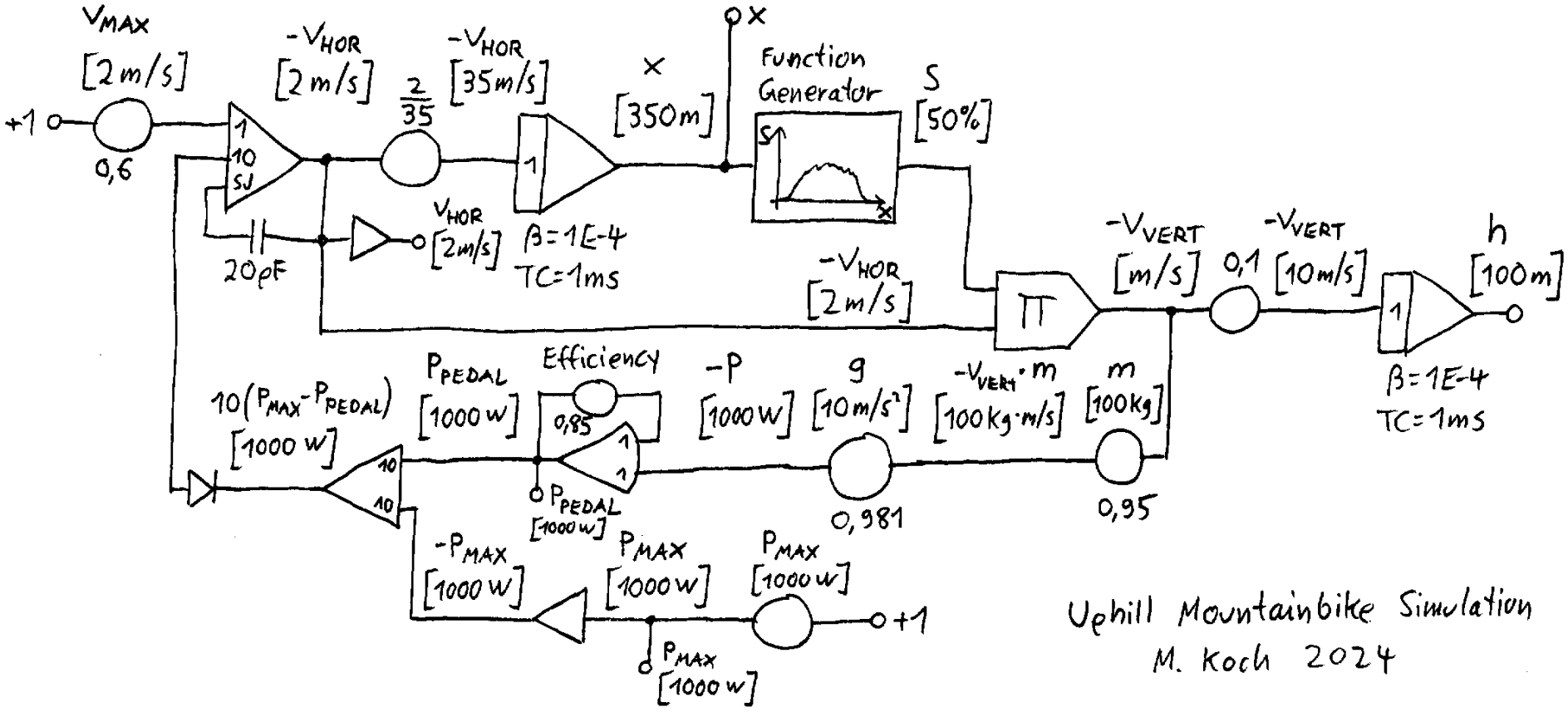
All signals should have a physical unit attached to them, which is normally written in [ ] brackets.

The resulting schematic diagram should be mathematically correct, but it won't run on the analog computer because it still has many problems:

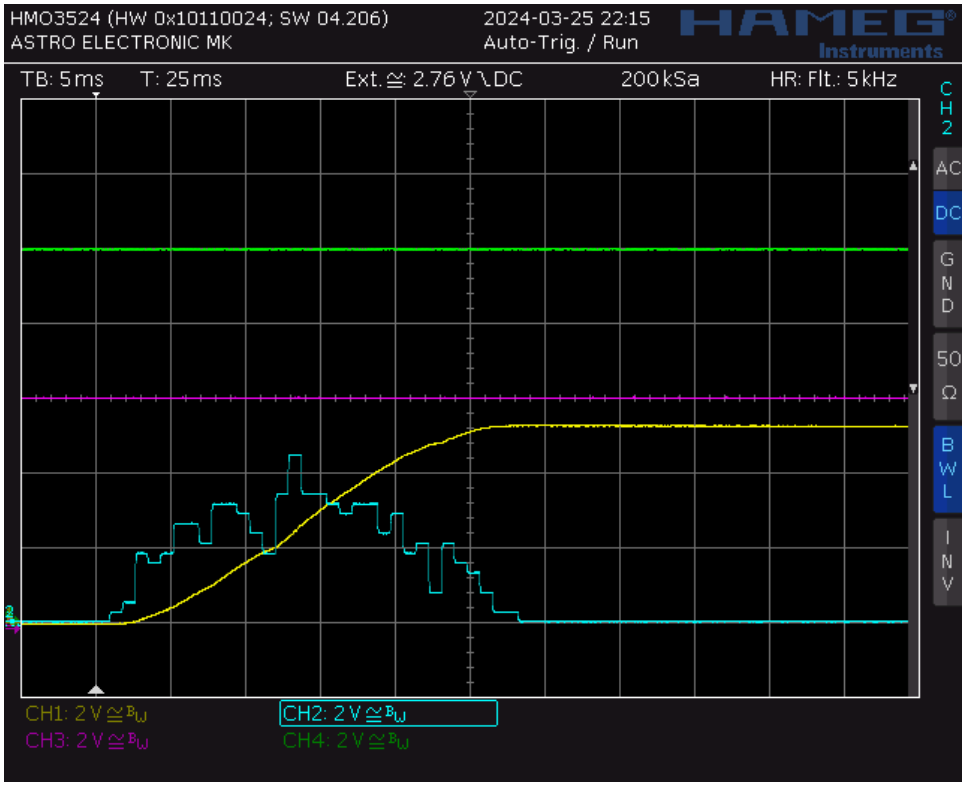
- **The time scale factor may be different from the desired value.**
- **The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode).**
- **The coefficients may be larger than +1 or smaller than -1 or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.05 and 1. In some cases 0.01 may be acceptable as well.**

Given is a velocity, which can be set with a coefficient from 0 to 2 m/s. The velocity is integrated to get the horizontal position x, which is the input signal for the digital function generator. The output of the function generator is the slope of the mountain. The slope is multiplied by the horizontal velocity to get the vertical velocity. Now the signal path splits. One path integrates the vertical velocity to get the elevation (and if all is correct, it must be 53 m at the end of the calculation). The other path calculates the required power, by multiplying with the total mass (cyclist and bicycle) and by gravitational acceleration. This is the theroretical power, but it's too small because there are also slippage losses between the rear wheel and the ground. This is modeled by dividing the power by a 0.85 efficiency constant. The result is the true power at the pedals. This power is compared with an adjustable maximum power. If the pedal power exceeds the maximum power, a penalty function becomes active (through the diode) and reduces the horizontal velocity. The simulation uses a time scale factor 1/10000, so that 300s in problem time are simulated in 30 ms in THAT.

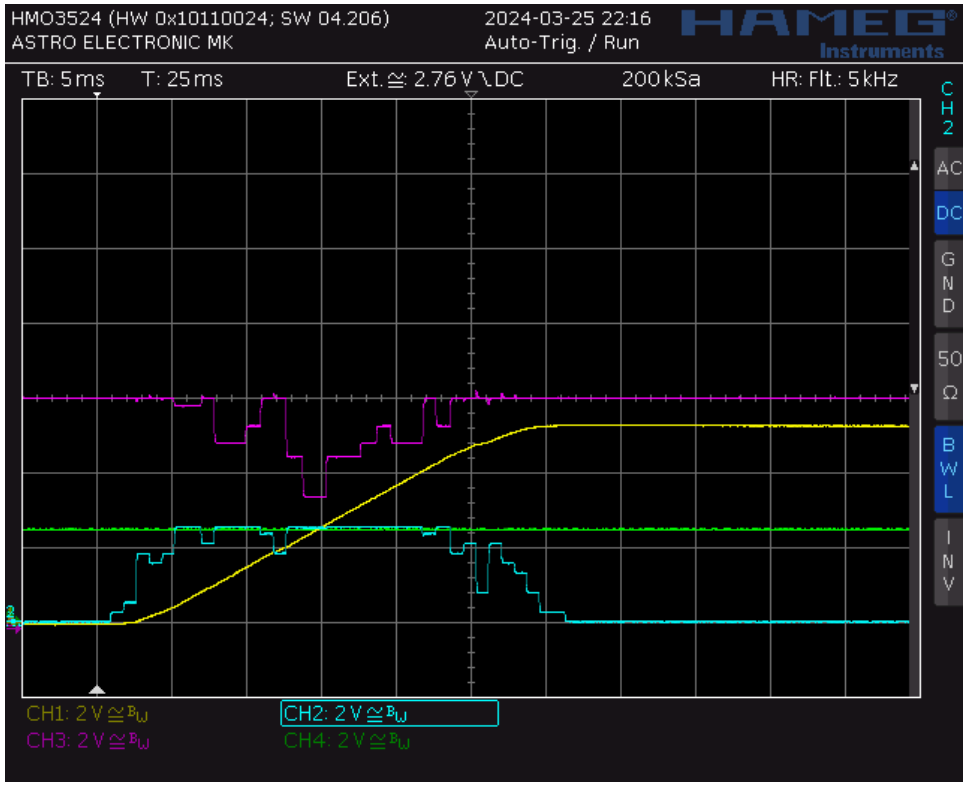
This is the circuit:



These are the results for V = 1.2 m/s, m = 95 kg, g = 9.81 m/s^2, Efficiency = 0.85.
Green: Power limit (200 W/div)
Cyan: Power at the pedals (200 W/div)
Magenta: Horizontal velocity (0.4 m/s / div)
Yellow: Elevation (20 m/div)



In this case the power limit (green) is set to 1000 W.
The peak pedal power (cyan) is about 460W, far below the limit.
That means the horizontal velocity (magenta) is always 1.2 m/s.
The elevation (yellow) goes from 0 to 53 m.
It takes about 28 ms (280s in problem time) to reach the peak of the mountain.



In this case the power limit (green) is set to 250 W, so that when the pedal power (cyan) exceeds the power limit, the horizontal velocity (magenta) is reduced.
It takes longer to reach the peak of the mountain, but the final elevation (yellow) is the same 53 m as in the left picture.
It takes about 34 ms (340s in problem time) to reach the peak of the mountain.

7.5 Falling Ball with Air Drag

A styrofoam ball with 80 mm diameter and 6.77 g mass falls down from 2 m height. It takes 0.68 s until it reaches the ground, as measured by a high speed camera. In this case, air drag has a significant influence.

Drag equation: https://en.wikipedia.org/wiki/Drag_equation

$$F_D = 0.5 \cdot \rho \cdot v^2 \cdot c_d \cdot A$$

with F_d = Drag force
- **ρ = Mass density of the fluid (1.25 kg / m³ for air)**
- **v = Flow velocity**
- **c_d = Dimensionless drag coefficient, see https://en.wikipedia.org/wiki/Drag_coefficient**
- **A = Reference area, here the cross section area of the ball, $\pi \cdot (0.04m)^2 = 0.00503\ m^2$**

The drag coefficient for a sphere is (collected from different sources):
- **0.47 for Re = 1e4**
- **0.45 for Re < 1.7e5**
- **0.09 to 0.18 for Re > 4.1e5**

See also (in german): https://de.wikipedia.org/wiki/Str%C3%B6mungswiderstandskoeffizient#cw-Werte_von_typischen_K%C3%B6rperformen

Re is the Reynolds number, which is defined as:

$$Re = \rho \cdot v \cdot L / \mu$$

with ρ = Mass density of the fluid (1.25 kg / m³ for air)
- **v = Flow velocity**
- **L = Characteristic length (here: diameter of the ball)**
- **μ = Dynamic viscosity of the fluid, here 18.2e-6 kg/(m·s) for air**

The drag coefficient of a sphere can be determined for the general case of a laminar flow using the following formula:
$C_d = 24 / Re + 4 / sqrt(Re) + 0.4$ **for Re < 200000 Source: https://en.wikipedia.org/wiki/Drag_(physics)#Low_Reynolds_Numbers:_Stokes'_drag**

Without air drag, the falling time would be $t = sqrt(2 \cdot h / a)$ which is 0.639 s, and the impact velocity would be $v_{max} = sqrt(2 \cdot g \cdot h)$ which is 6.26 m/s. With this velocity the Reynolds number for the ball with 0.08 m diameter is 34396, so that the drag coefficient should be 0.42.

The idea is to make a simulation on the analog computer, with adjustable drag coefficient from 0 to 1. Without air drag the time should be 0.639 s, and with the drag coefficient set to 0.42 the time should be 0.68 s.

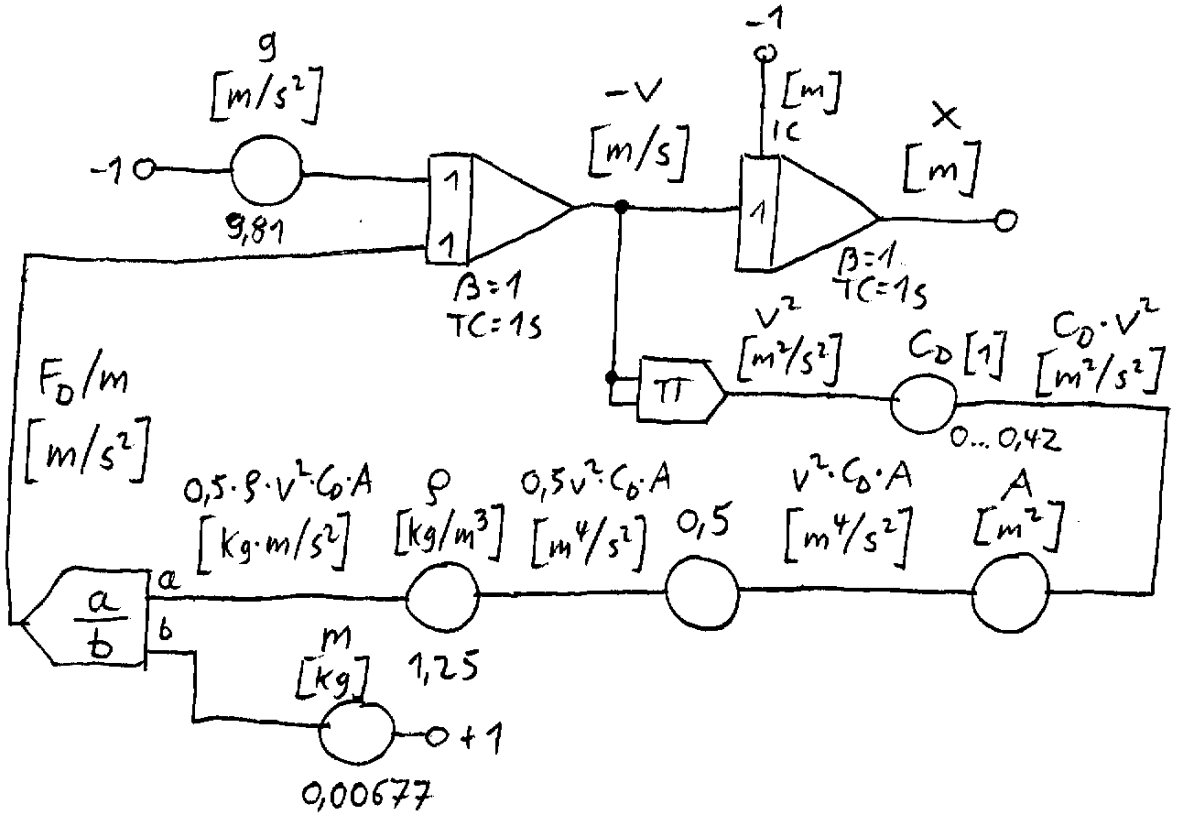
Let's write down the equations:

Velocity: $v = dx / dt$
Acceleration: $a = dv / dt$
There are two forces acting on the ball (positive is up):

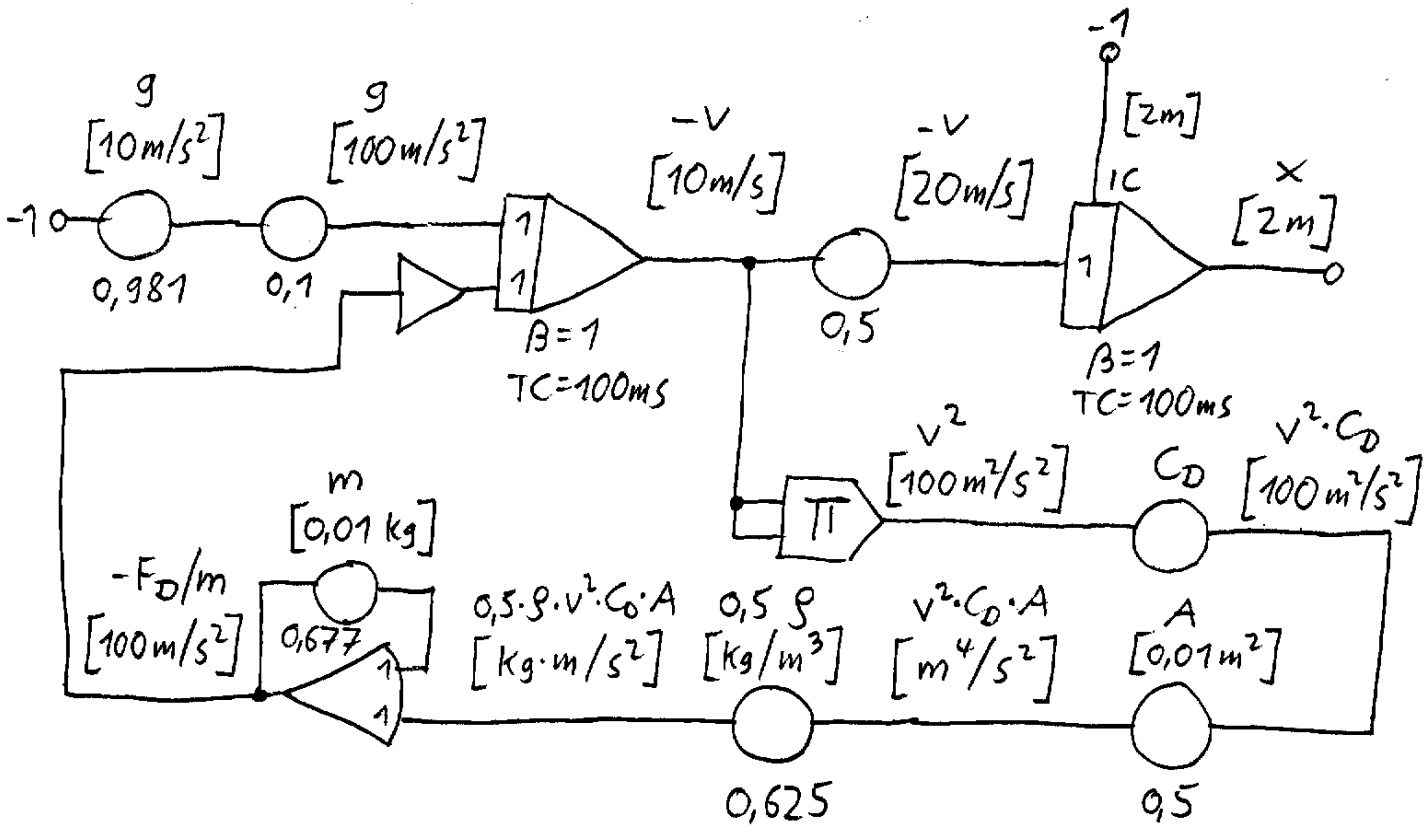
- **Weight force:** $F_g = -m \cdot g$
- **Air drag force:** $F_d = 0.5 \cdot \rho \cdot v^2 \cdot c_d \cdot A$

These two forces are acting in opposite directions.
Total force: $F = F_d - F_g$
Total acceleration (positive up): $a = F_d / m - g$
Initial conditions: $x = 2$ m $\quad v = 0$ m/s
Time scale factor: $\beta = 1$ (no time scaling)

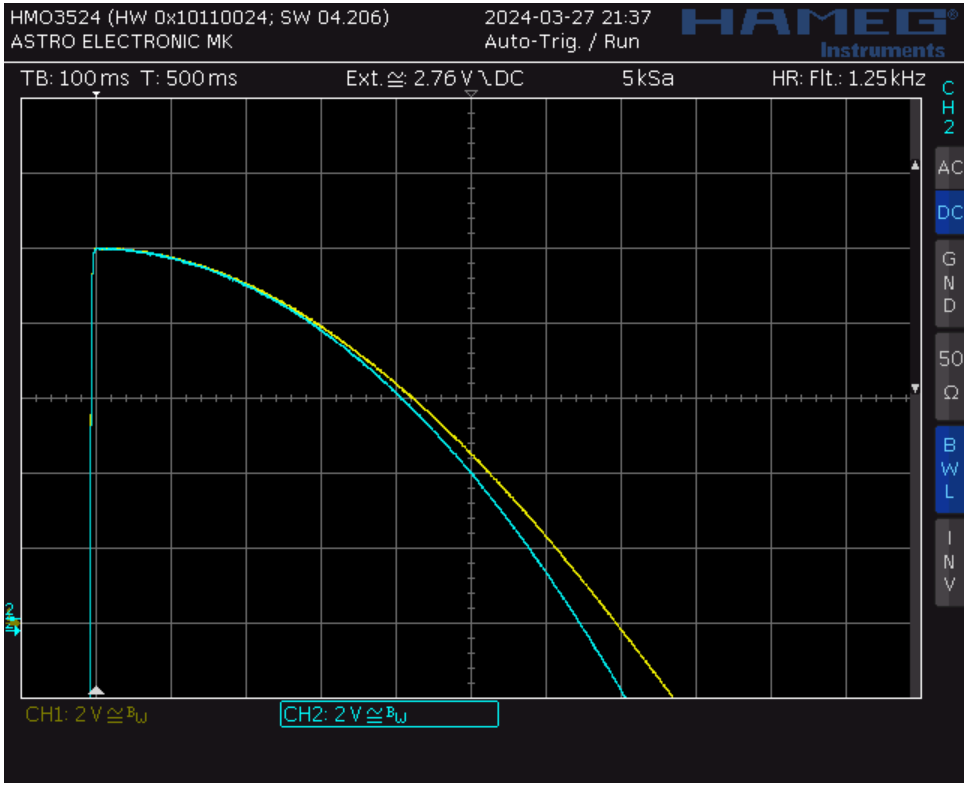
This is the circuit before amplitude scaling:



This is the circuit after amplitude scaling (80mm ball, 2 m height, coefficient C_D is 0.42):

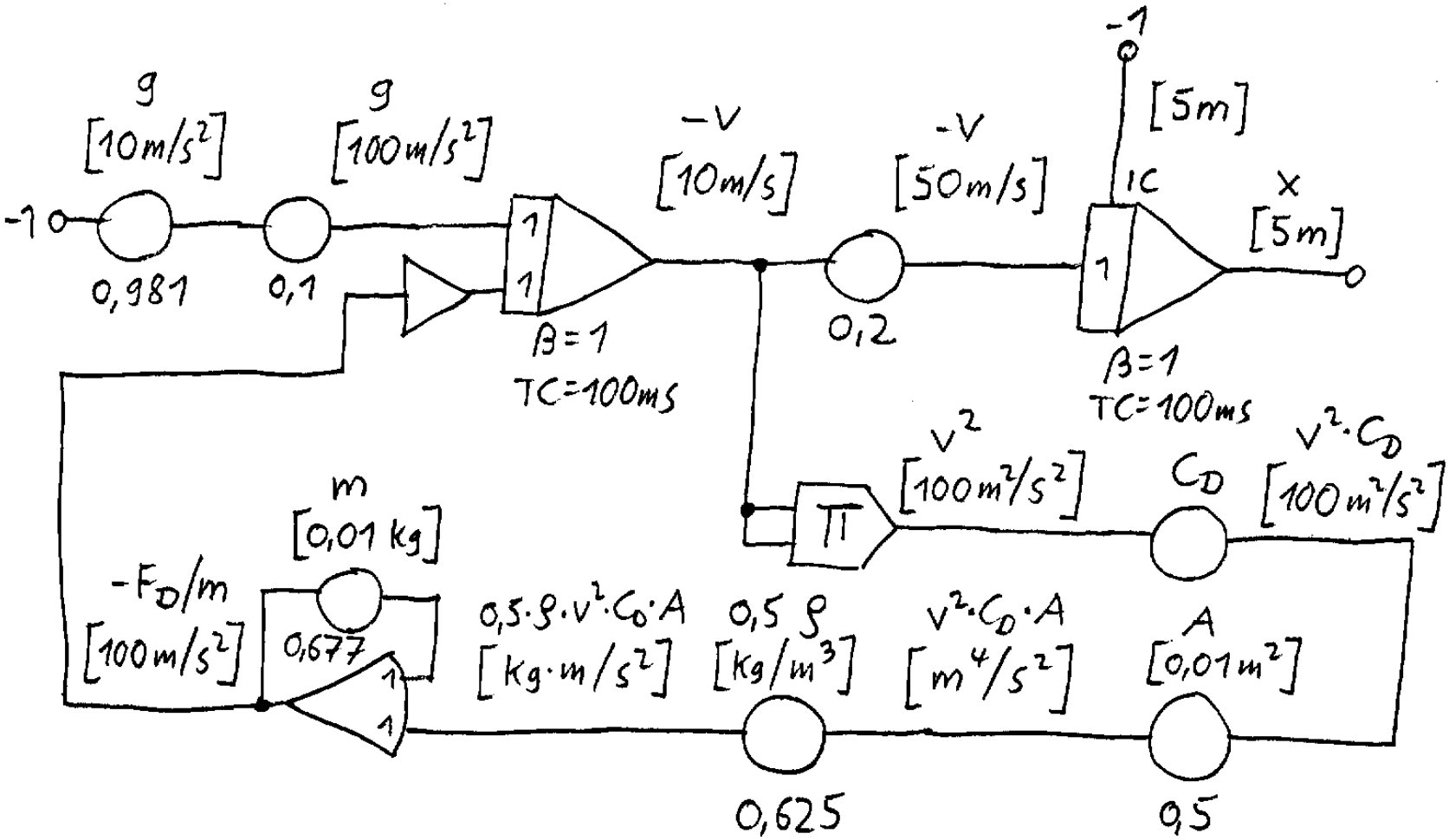


This is the result of the calculation (80mm ball, 2 m height). Cyan is without air drag, and yellow is with C_D = 0.42 air drag, which takes about 50 ms longer.



See also (in german): https://www.dlr.de/schoollab/Portaldata/24/Resources/dokumente/go/14.8.02_Ankuendigung.pdf
https://www.unterrichtsmodule-energie.de/forscherauftrag-juni-2021/

This is the same circuit as before for the 80 mm ball, but the height is changed to 5 m. Only the coefficient at the input of the position integrator must be changed from 0.5 to 0.2. The expected velocity is still in the allowed range, just below 10 m/s without air drag. Coefficient C_D is 0.42.

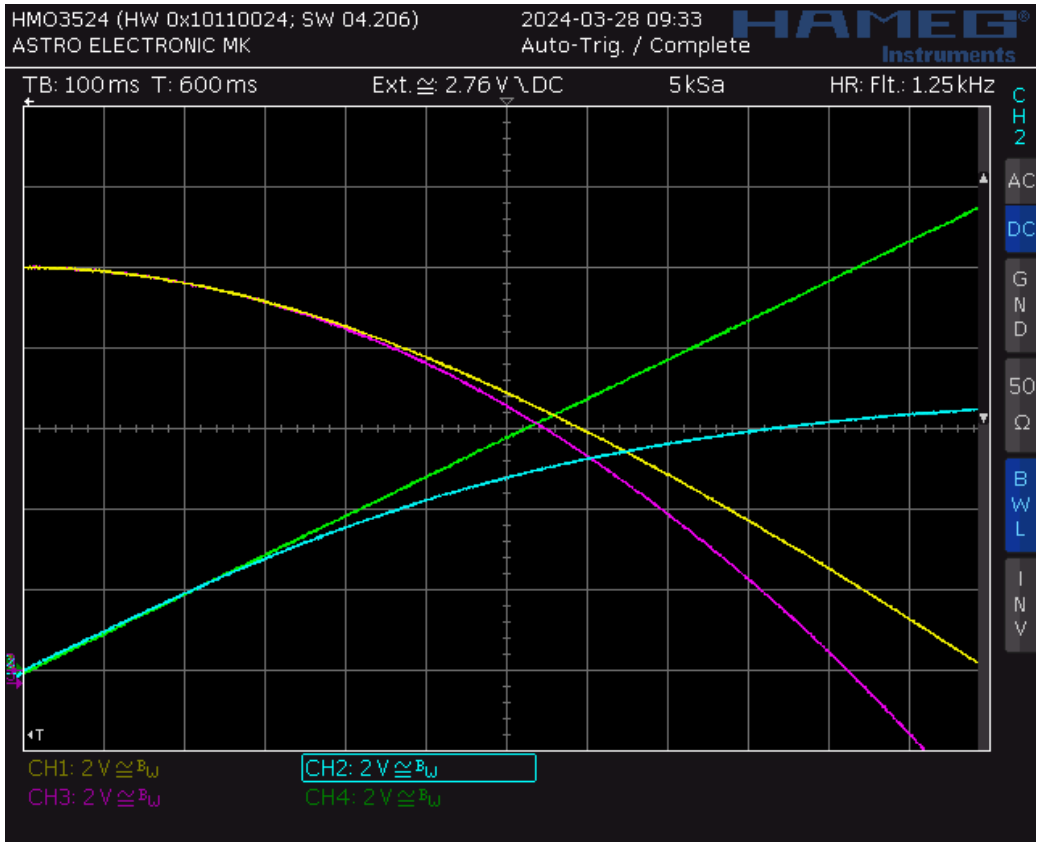


This is the output for the 80 mm ball and 5 m falling height.

Yellow: Position with air drag
Cyan: Inverted velocity with air drag

Magenta: Position without air drag
Green: Inverted velocity without air drag

The measured time for 5 m height is 1.215 s, which is in very good agreement with the simulation.



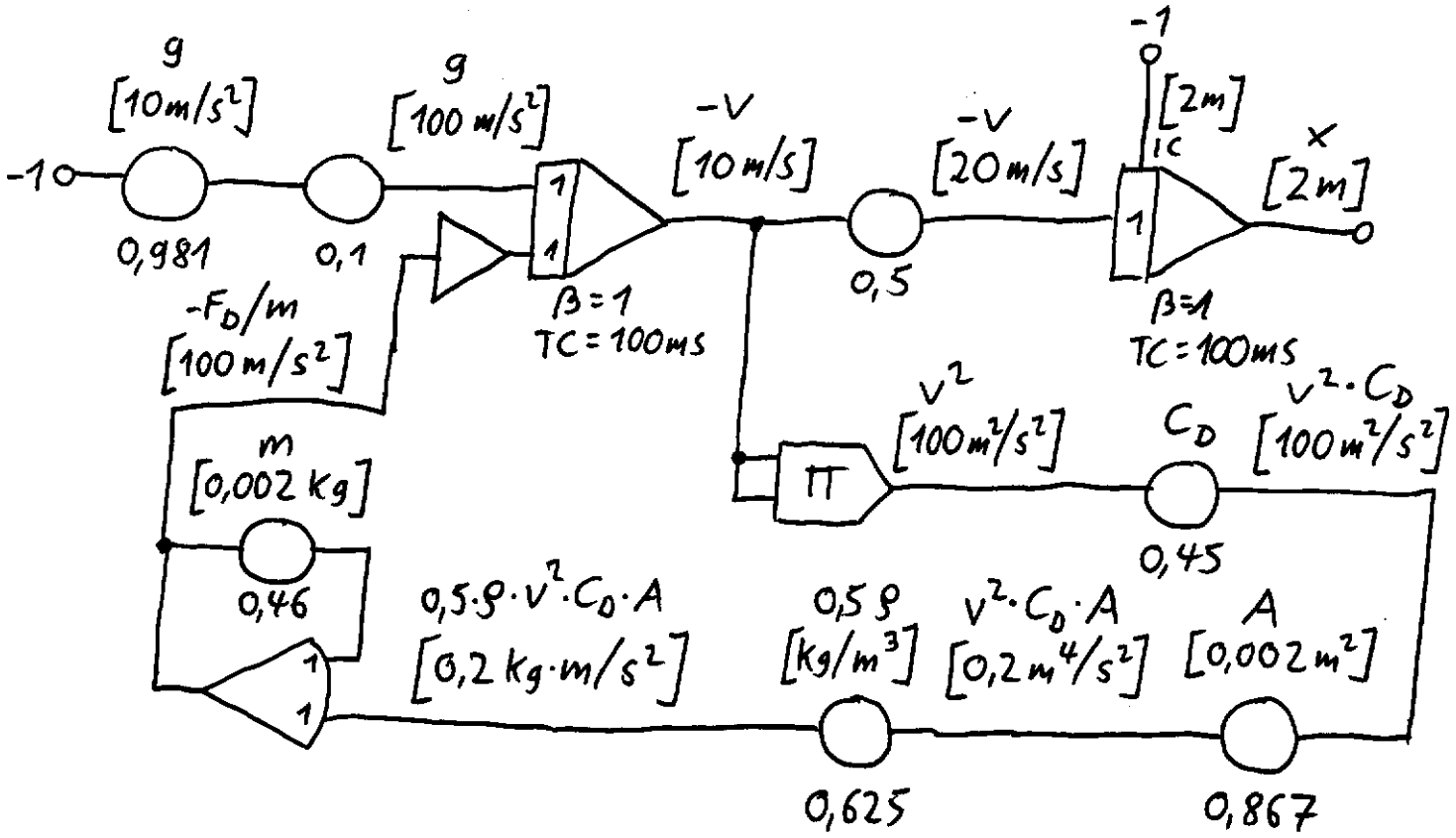
Lets'repeat the experiment with a smaller styrofoam ball with 47 mm diameter and 0.92 g mass, falling down from 2 m height. The smaller ball should be even more affected by air drag.

Without air drag, the falling time would be $t = \sqrt{2 \cdot h / a}$ which is 0.639 s, and the impact velocity would be $v_{max} = \sqrt{2 \cdot g \cdot h}$ which is 6.26 m/s.

The Reynolds number is: $Re = \rho \cdot v \cdot L / \mu = 1.25\ kg / m^3 \cdot 6.26\ m / s \cdot 0.047\ m / 18.2e\text{-}6\ kg/(m \cdot s) = 7432$

Using the formula for the drag coefficient of a sphere, $C_d = 24 / Re + 4 / \sqrt{Re} + 0.4$ for $Re < 200000$ we get $C_d = 0.45$

This is the circuit for the 47mm ball falling from 2 m height:

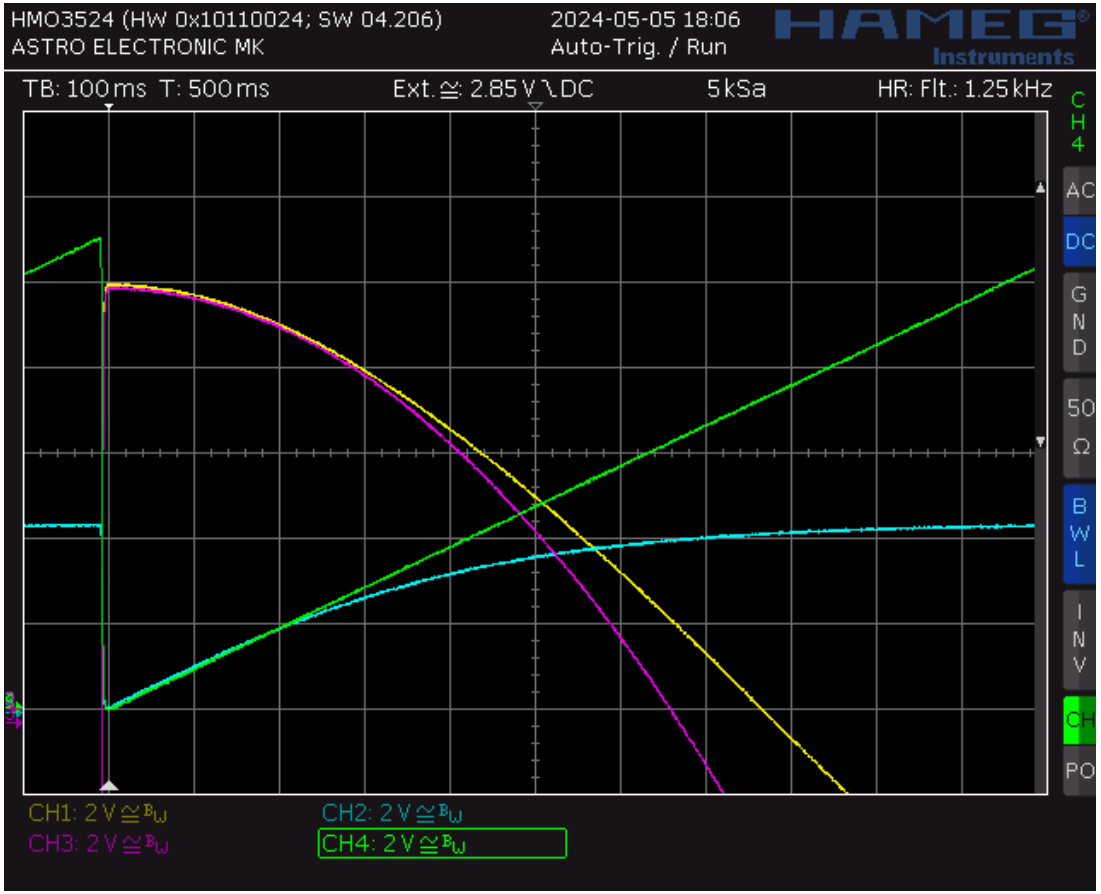


This is the output for the 47mm ball falling from 2 m height.

Yellow: Position with air drag
Cyan: Inverted velocity with air drag

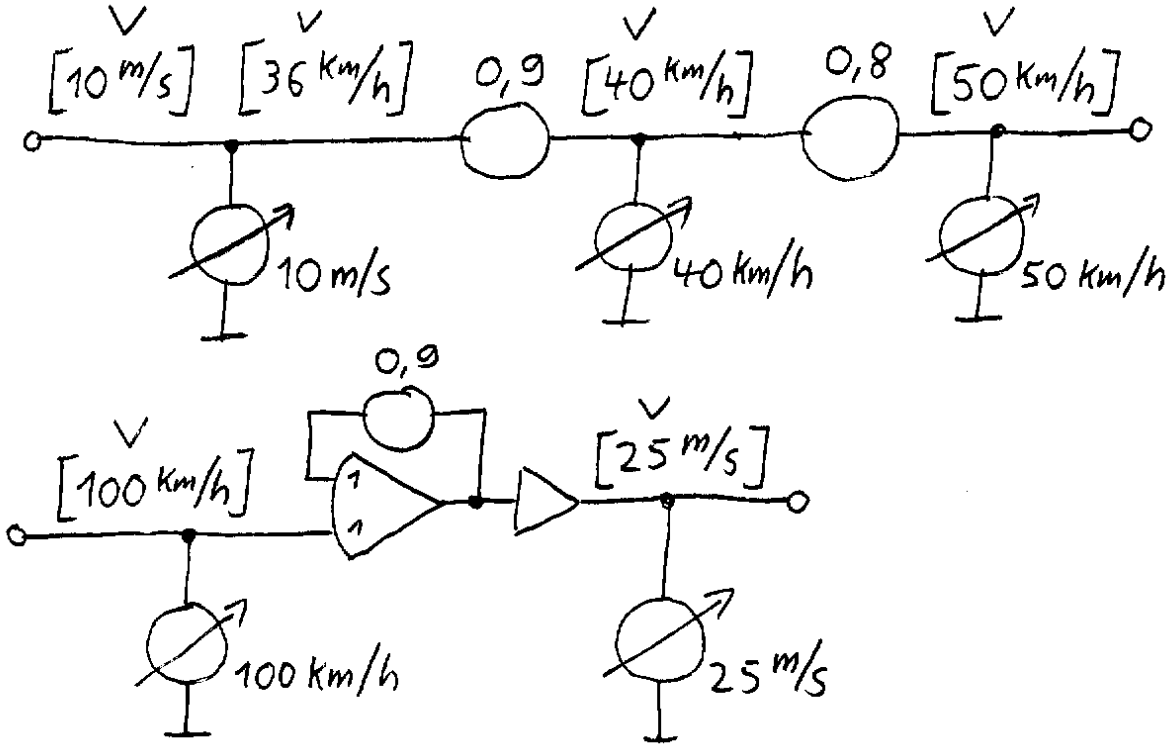
Magenta: Position without air drag
Green: Inverted velocity without air drag

Calculated time with air drag: about 0.77 s



7.6 Unit Conversion for Velocity m/s ↔ km/h

m/s	km/h
10	36
11.111	40
12	43.2
13.888	50
15	54
20	72
25	90
27.777	100
40	144
50	180
55.555	200
80	288
100	360



7.7 Vehicle Simulation, Part 1: Power

The required power of a vehicle (car, e-bike, bicycle) driving at constant velocity consists of these three components:

$P = P_{SLOPE} + P_{ROLL} + P_{DRAG}$

$P_{SLOPE} = v \cdot slope \cdot m \cdot g$ Slope of the road

$P_{ROLL} = v \cdot c_{rr} \cdot m \cdot g$ Rolling resistance

$P_{DRAG} = v^3 \cdot 0.5 \cdot \rho \cdot c_d \cdot A$ Air drag

with v = velocity
slope = Dimensionless slope (1 means 45°)
m = Total mass of the vehicle
g = Gravitational acceleration 9.81 m/s^2
c_{rr} = Dimensionless rolling resistance coefficient, see https://en.wikipedia.org/wiki/Rolling_resistance
ρ = Air density 1.25 kg/m^3
c_d = Dimensionless drag coefficient, see https://en.wikipedia.org/wiki/Drag_coefficient
A = Reference area, here the front area of the vehicle

Alternatively the *drag area* $c_d \cdot A$ can be specified.

Obviously the formulas for P_{SLOPE} and P_{ROLL} are quite similar and can easily be combined into one formula:

$P_{SLOPE\_ROLL} = v \cdot slope_{rr} \cdot m \cdot g$ with $slope_{rr} = slope + c_{rr}$

Why is the drag power proportional to v^3 ?

Because drag force is proportional to v^2: $F_d = 0.5 \cdot \rho \cdot v^2 \cdot c_d \cdot A$
and drag power is proportional to v: $P_d = F_d \cdot v$

Typical values for a few vehicles:

Vehicle	max. Power P [W]	Total mass (including driver) m [kg]	Front area A [m^2]	Drag coefficient c_d	Drag area $c_d \cdot A$ [m^2]	Rolling resistance coefficient c_{rr}
Mountainbike	about 400	95			0.59	0.0046
Racing bike	about 400	90			0.35	0.0033
VW Caddy	55000	1550 - 2035	2.83	0.35	0.991	0.011 - 0.015
Lamborghini Countach	335000	1525	1.76	0.40	0.704	0.011 - 0.015

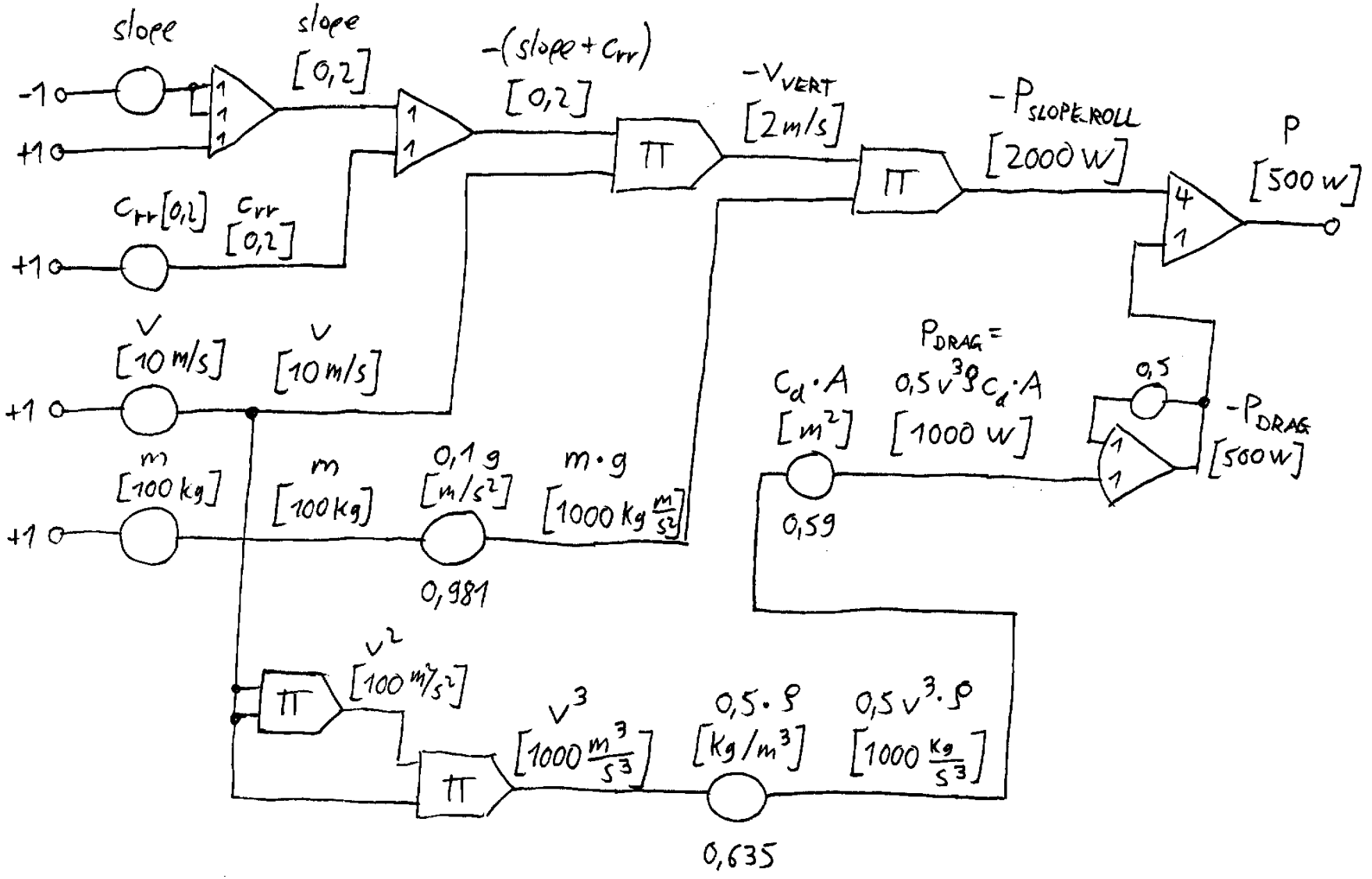
Source for drag area for bicycles (german): http://www.kreuzotter.de/deutsch/speed.htm

Source for drag coefficient c_d and area A for cars (german): https://automobil-guru.de/cw-werte-tabelle-stirnflaeche/ http://rc.opelgt.org/indexcw.php

Source for rolling resistance coefficient c_{rr} (english): https://en.wikipedia.org/wiki/Rolling_resistance

Source for rolling resistance coefficient c_{rr} (german): https://de.wikipedia.org/wiki/Rollwiderstand

This is the circuit, with the parameters optimized for mountainbike:



7.8 Vehicle Simulation, Part 2: Add Wind Velocity

7.9 Vehicle Simulation, Part 3: Energy

Great software for creating elevation profiles: https://brouter.de/brouter-web/

Comparison of the same 2.1 km geographic track with elevation data, in three different file formats:

***.gpx 30 waypoints, 2484 bytes**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- track-length = 2098 filtered ascend = 47 plain-ascend = 44 cost=3235 energy=0.0kwh time=10m 36s -->
<gpx
 xmlns="http://www.topografix.com/GPX/1/1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX/1/1/gpx.xsd"
 version="1.1"
 creator="BRouter-Web 0.18.1">
 <trk>
  <name>xx (2,1 km)</name>
  <trkseg>
   <trkpt lat="51.648204" lon="10.344337"><ele>255.25</ele></trkpt>
   <trkpt lat="51.647634" lon="10.342991"><ele>255</ele></trkpt>
   <trkpt lat="51.647455" lon="10.341876"><ele>254.25</ele></trkpt>
   <trkpt lat="51.647213" lon="10.340211"><ele>247.75</ele></trkpt>
   <trkpt lat="51.645843" lon="10.340745"><ele>253.75</ele></trkpt>
   <trkpt lat="51.644352" lon="10.341321"><ele>251.5</ele></trkpt>
   <trkpt lat="51.64476" lon="10.344264"><ele>256.25</ele></trkpt>
   <trkpt lat="51.645181" lon="10.347296"><ele>256.25</ele></trkpt>
   <trkpt lat="51.645204" lon="10.347461"><ele>256.25</ele></trkpt>
   <trkpt lat="51.645578" lon="10.35003"><ele>258.25</ele></trkpt>
   <trkpt lat="51.645637" lon="10.350244"><ele>258.75</ele></trkpt>
   <trkpt lat="51.645744" lon="10.350452"><ele>259.5</ele></trkpt>
   <trkpt lat="51.645814" lon="10.350561"><ele>259.75</ele></trkpt>
   <trkpt lat="51.646103" lon="10.351093"><ele>261.25</ele></trkpt>
   <trkpt lat="51.646189" lon="10.351247"><ele>262</ele></trkpt>
   <trkpt lat="51.646301" lon="10.351473"><ele>263.25</ele></trkpt>
   <trkpt lat="51.646427" lon="10.351731"><ele>264.75</ele></trkpt>
   <trkpt lat="51.646627" lon="10.352792"><ele>268.75</ele></trkpt>
   <trkpt lat="51.646953" lon="10.354453"><ele>274.5</ele></trkpt>
   <trkpt lat="51.647011" lon="10.35471"><ele>275.5</ele></trkpt>
   <trkpt lat="51.647074" lon="10.354879"><ele>276.5</ele></trkpt>
   <trkpt lat="51.647528" lon="10.355727"><ele>281.5</ele></trkpt>
   <trkpt lat="51.648405" lon="10.35734"><ele>289.75</ele></trkpt>
   <trkpt lat="51.648544" lon="10.358155"><ele>292.5</ele></trkpt>
   <trkpt lat="51.648631" lon="10.358745"><ele>293.25</ele></trkpt>
   <trkpt lat="51.648628" lon="10.358907"><ele>293.5</ele></trkpt>
   <trkpt lat="51.648595" lon="10.35907"><ele>294</ele></trkpt>
   <trkpt lat="51.648508" lon="10.359226"><ele>294.75</ele></trkpt>
   <trkpt lat="51.647886" lon="10.360043"><ele>299.75</ele></trkpt>
   <trkpt lat="51.647851" lon="10.36009"><ele>296.25</ele></trkpt>
  </trkseg>
 </trk>
</gpx>
```

***.kml 30 waypoints, 1127 bytes, I think this is the best format**

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
 <Document>
  <Placemark>
   <name>xx (2,1 km)</name>
   <ExtendedData>
    <Data name="name">
     <value>xx (2,1 km)</value>
    </Data>
   </ExtendedData>
   <LineString>
    <coordinates>10.344337,51.648204,255.25 10.342991,51.647634,255 10.341876,51.647455,254.25 10.340211,51.647213,247.75 10.340745,51.645843,253.75 10.341321,51.644352,251.5 10.344264,51.64476,256.25
10.347296,51.645181,256.25 10.347461,51.645204,256.25 10.35003,51.645578,258.25 10.350244,51.645637,258.75 10.350452,51.645744,259.5 10.350561,51.645814,259.75 10.351093,51.646103,261.25
10.351247,51.646189,262 10.351473,51.646301,263.25 10.351731,51.646427,264.75 10.352792,51.646627,268.75 10.354453,51.646953,274.5 10.35471,51.647011,275.5 10.354879,51.647074,276.5
10.355727,51.647528,281.5 10.35734,51.648405,289.75 10.358155,51.648544,292.5 10.358745,51.648631,293.25 10.358907,51.648628,293.5 10.35907,51.648595,294 10.359226,51.648508,294.75
10.360043,51.647886,299.75 10.36009,51.647851,296.25</coordinates>
   </LineString>
  </Placemark>
 </Document>
</kml>
```

***.csv 10 waypoints, 1148 bytes**

Longitude	Latitude	Elevation	Distance	CostPerKm	ElevCost	TurnCost	NodeCost	InitialCost	WayTags	NodeTags	Time	Energy
10340211	51647213	247	311	1150	0	5	0	0	highway=residential		41	4191
10341321	51644352	251	328	1050	0	90	0	0	highway=track tracktype=grade1 surface=asphalt		98	9888
10347296	51645181	256	209	1150	0	88	0	0	highway=track tracktype=grade2 bicycle=yes vehicle=no		142	14278
10347296	51645181	256	215	1150	0	0	0	0	highway=track tracktype=grade2 bicycle=yes vehicle=no	railway=level_crossing	48	4813
10350452	51645744	259	230	1150	0	4	0	0	highway=track tracktype=grade2 bicycle=yes vehicle=no		99	9939
10350561	51645814	259	11	1050	0	0	0	0	highway=track tracktype=grade1 surface=asphalt		102	10213
10351247	51646189	262	63	1050	0	0	0	0	reversedirection=yes highway=track tracktype=grade1 surface=asphalt		121	12168
10351473	51646301	263	20	1350	0	0	0	0	reversedirection=yes highway=track tracktype=grade1		125	12540
10355727	51647528	281	331	2050	0	8	0	0	reversedirection=yes highway=track tracktype=grade2		288	28809
10360828	51647312	296	380	2050	0	23	0	0	reversedirection=yes highway=track tracktype=grade2 surface=compacted		492	49295

Online tool for making mountain panoramas: https://www.udeuschle.de/panoramas/makepanoramas.htm

7.10 Falling tilted Bar

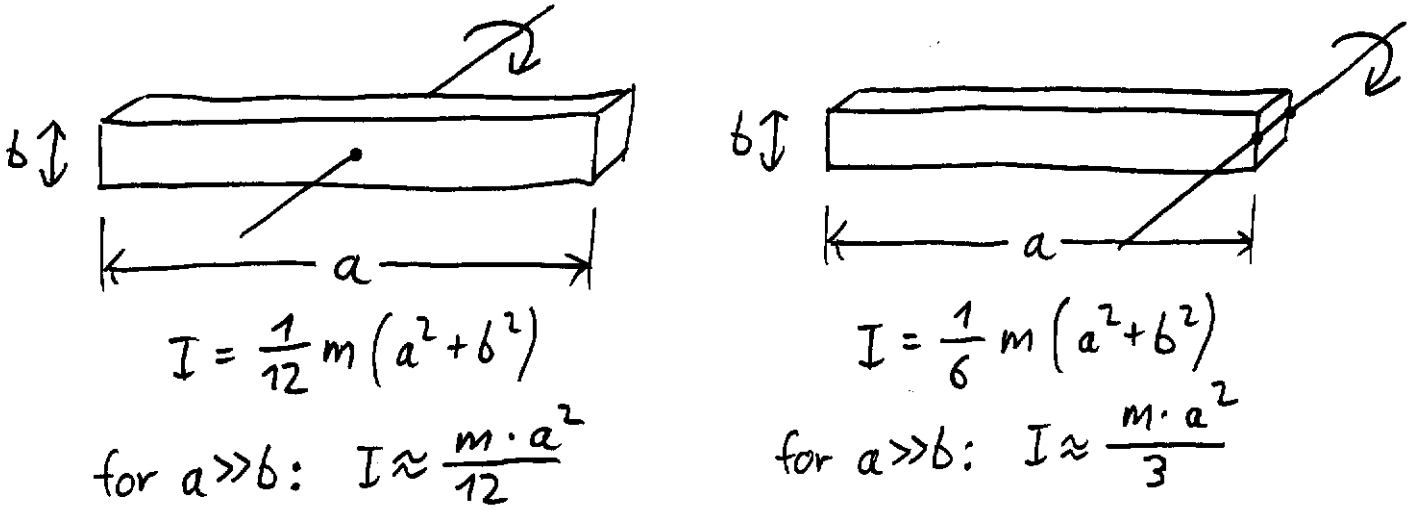
Simulate a stiff tilted bar, falling down on a surface, as a simplified case of the falling rope ladder:

https://www.facebook.com/watch?v=536396402344803

http://ruina.tam.cornell.edu/research/topics/fallingchains/Falling_Chain_experiments.html

http://ruina.tam.cornell.edu/research/topics/fallingchains/chain_paperV13revised.pdf

Moment of inertia of a bar with mass m , for two different rotation axes:

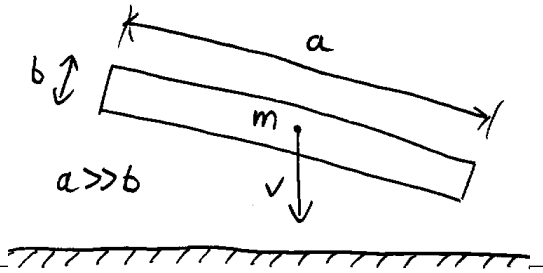


Other case, not relevant here: If all mass is concentrated at the ends of the bar (m/2 at each end), then the moment of inertia is:

$I = m / 4 \cdot a^2$ (for rotation axis in the center)

Rotational (or angular) movement		Translational (or linear) movement	
Moment of intertia https://en.wikipedia.org/wiki/Moment_of_inertia (german: Drehmoment)	I (or θ) [kg m^2]	Mass (german: Masse)	m [kg]
Angular velocity (Angular frequency, Angular speed, Angular rate) https://en.wikipedia.org/wiki/Angular_velocity (german: Rotationsgeschwindigkeit, Drehgeschwindigkeit)	ω [rad s^{-1}] [s^{-1}] Linear velocity at radius r: $v = \omega \cdot r$	Velocity (german: Geschwindigkeit)	$v = s / t$ [m s^{-1}] Angular frequency at radius r: $\omega = v / r$
Angular acceleration https://en.wikipedia.org/wiki/Angular_acceleration (german: Winkelbeschleunigung)	$\alpha = \omega / t$ [rad s^{-2}] [s^{-2}]	Acceleration (german: Beschleunigung)	$a = v / t$ [m s^{-2}]
Torque (Moment, Moment of force) https://en.wikipedia.org/wiki/Torque (german: Drehmoment)	$M = I \cdot \alpha$ [Nm]	Force (german: Kraft)	$F = m \cdot a$ [N]
Angular momentum (Moment of momentum, Rotational momentum) https://en.wikipedia.org/wiki/Angular_momentum (german: Drehimpuls) Balance of angular momentum: https://en.wikipedia.org/wiki/Balance_of_angular_momentum	$L = I \cdot \omega$ [kg m^2 s^{-1} = Nms] $L = r \times p$ where r is the radius from the center of gravity to the point where momentum p is applied, If the angle is 90°, the cross product × can be replaced by normal multiplication.	Momentum (Linear momentum, Translational momentum) https://en.wikipedia.org/wiki/Momentum https://en.wikipedia.org/wiki/Impulse_(physics) (german: Impuls)	$p = m \cdot v$ [kg m s^{-1} = Ns]
Rotational energy https://en.wikipedia.org/wiki/Rotational_energy (german: Energie)	$E_{ROT} = 0.5 \cdot I \cdot \omega^2$ [Nm = Ws]	Energy (german: Energie)	$E_{KIN} = 0.5 \cdot m \cdot v^2$ [Nm = Ws]

A tilted bar with mass m is falling down to a flat surface with velocity v. The tilt angle is small and the right edge hits the surface first. Let's assume the cross section b is small compared to the length a. The center of gravity is in the middle of the bar, and the mass is uniformly distributed along the bar.



Mass m = 1 kg
Length a = 1 m
Velocity v = 1 m/s (positive downwards)
Moment of inertia I = 0.0833 kg m^2 (Rotation axis in the middle of the bar)

Assumption: Elastic collision, the sum of translational and rotational energy remains constant.

Case	Translational velocity (at center of mass)	Translational energy $E_{KIN} = 0.5 \cdot m \cdot v^2$	Rotational velocity	Rotational energy $E_{ROT} = 0.5 \cdot I \cdot \omega^2$	Velocity at the edges of the bar, due to rotation $v_{ROT} = \omega \cdot 0.5 \cdot a$		Velocity at the edges of the bar, sum of translational and rotational velocity		
					Left edge	Right edge	Left edge	Center	Right edge
Just before collision, 100% translational energy	v = 1 m/s	E_{KIN} = 0.5 Nm	ω = 0 s^{-1}	E_{ROT} = 0 Nm	0 m/s	0 m/s	1 m/s	1 m/s	1 m/s
75% translational energy, 25% rotational energy, The right edge sticks to the surface	v = 0.866 m/s	E_{KIN} = 0.375 Nm	ω = 1.732 s^{-1}	E_{ROT} = 0.125 Nm	0.866 m/s	-0.866 m/s	1.732 m/s (*)	0.866 m/s	0 m/s
50% translational energy, 50% rotational energy	v = 0.707 m/s	E_{KIN} = 0.25 Nm	ω = 2.449 s^{-1}	E_{ROT} = 0.25 Nm	1.225 m/s	-1.225 m/s	1.932 m/s	0.707 m/s	-0.518 m/s
33.3% translational energy, 66.7% rotational energy	v = 0.577 m/s	E_{KIN} = 0.167 Nm	ω = 2.828 s^{-1}	E_{ROT} = 0.333 Nm	1.414 m/s	-1.414 m/s	1.991 m/2	0.577 m/s	-0.837 m/s
25% translational energy, 75% rotational energy	v = 0.5 m/s	E_{KIN} = 0.125 Nm	ω = 3 s^{-1}	E_{ROT} = 0.375 Nm	1.5 m/s	-1.5 m/s	2 m/s	0.5 m/s	-1 m/s
100% rotational energy	v = 0 m/s	E_{KIN} = 0 Nm	ω = 3.464 s^{-1}	E_{ROT} = 0.5 Nm	1.732 m/s	-1.732 m/s	1.732 m/s	0 m/s	-1.732 m/s

(*) This is wrong. In this paper http://ruina.tam.cornell.edu/research/topics/fallingchains/Falling_Chain_experiments.html
the author claims in formula (9) that the velovity is 1.5 m/s

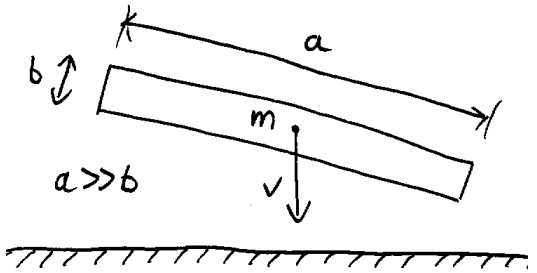
The above calculation assumes elastic collision where energy is conserved. That's wrong.
Instead we must assume an inelastic collision, where momentum is conserved.

Now let's try to solve the problem, assuming an inelastic collision:

https://en.wikipedia.org/wiki/Inelastic_collision

In an inelastic collision, the (linear or angular) momentum is conserved.

A tilted bar with mass m is falling down to a flat surface with velocity v. The tilt angle is small and the right edge hits the surface first. Let's assume the cross section b is small compared to the length a. The center of gravity is in the middle of the bar, and the mass is uniformly distributed along the bar.



Mass m = 1 kg
Length a = 1 m
Velocity v = 1 m/s (positive downwards)
Moment of inertia $I = 0.333\ kg\ m^2$ (Rotation axis at the edge of the bar)

The momentum of the falling bar is $p = m \cdot v = 1\ Ns$

When the right edge touches the surface, half of the momentum is transferred to the surface: $p_1 = 0.5\ Ns$. (If the center of the bar would touch the surface, or if both ends would touch the surface simultaneously, then the whole momentum would be transferred).

This momentum is immediately transferred back to the bar, but now as an angular momentum.

Angular momentum can be described as $L = I \cdot \omega$ or as $L = r \times p_1$ where r is the radius from the center of mass to the point where momentum p is applied. If the momentum p is applied rectangular to the radius vector, then the cross product can be simplified as a normal product.

In this case r = a and $I = 0.333\ kg\ m^2$.

So we get this equation:

$I \cdot \omega = r \cdot p_1$ or $\omega = r \cdot p_1 / I$

$\omega = 1\ m \cdot 0.5\ Ns / (0.333\ kg\ m^2) = 1.5\ rad/s$

To get v_{ROT} at the left edge of the bar, we must multiply ω by the length of the bar:

$v_{ROT} = \omega * a = 1.5\ m/s$

Many thanks to Dr. Roland Kruse, TU Braunschweig, Institut für angewandte Mechanik, who gave me the following solution:

It's sufficient to use conservation of angular momentum (relative to the contact point at the right edge):

Before collision: $L_{BEFORE} = m \cdot v \cdot a/2$

After collision, rotation with ω around the contact point:

$L_{AFTER} = \theta \cdot \omega$ **where θ is the moment of intertia around the contact point,** $\theta = 1/3 \cdot m \cdot a^2 = 0.333\ kg\ m^2$

with $v_1 = \omega \cdot a$ **or** $\omega = v_1 / a$

$L_{AFTER} = 1/3 \cdot m \cdot a^2 \cdot v_1 / a$

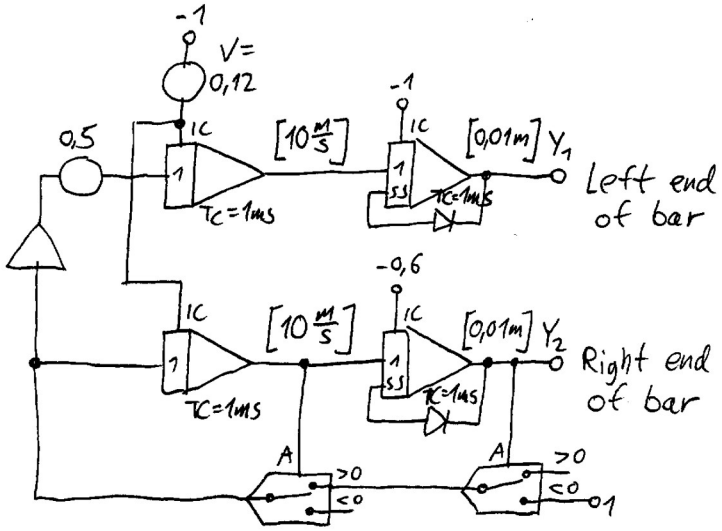
$L_{AFTER} = 1/3 \cdot m \cdot v_1 \cdot a$

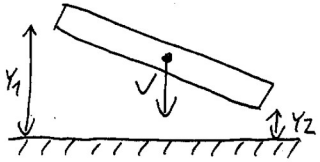
$L_{BEFORE} = L_{AFTER}$

$m \cdot v \cdot a/2 = 1/3 \cdot m \cdot v_1 \cdot a$

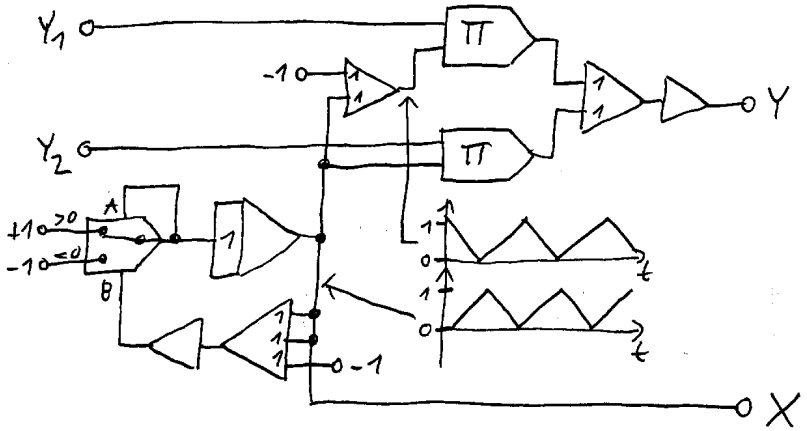
$v_1 = 3/2 \cdot v$

Circuit for simulation of the falling bar:



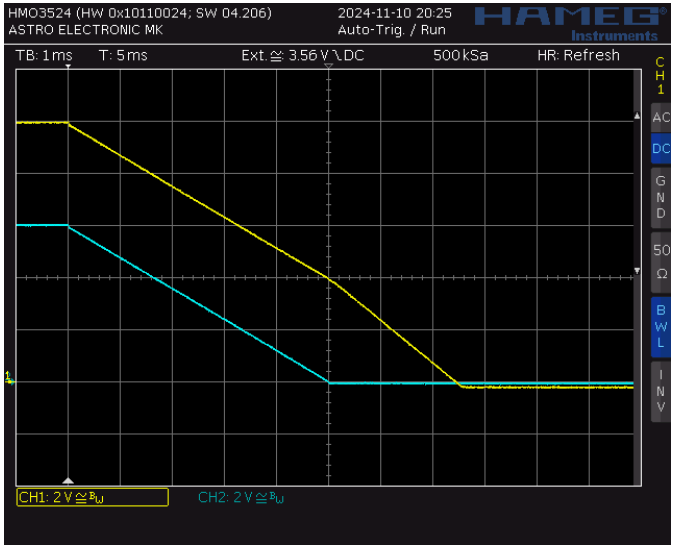


Circuit for visualizing the bar on an oscilloscope with XY display:



Screenshot from oscilloscope, Y1 and Y2 oder time:

When the right end (cyan) touches the ground, the left end (yellow) becomes faster by a factor 3/2.



7.11 Ideas for more Examples

- **Entry of a meteor into the atmosphere. Parameters are velocity, angle, mass, diameter. Calculate how fast it burns up in the atmosphere, and if it reaches the ground or not, or calculate the diameter of the crater.**

  See also (in german): Rendtel, Jürgen and Arlt, Rainer: Meteore
  Page 27: Entry velocity is between 11 and 73 km/s, first traces of ionisation are visible in 140 km height
  Page 28: Several formulas for brightness
  Page 29: Strong breaking process begins below 60 km height

- **Lunar landing as a quantitative calculation.**

- **Light path around a black hole? Schwarzschild radius: $r_s = 2 \cdot G \cdot m / c^2$**
  https://en.wikipedia.org/wiki/Event_horizon
  https://de.wikipedia.org/wiki/Ereignishorizont

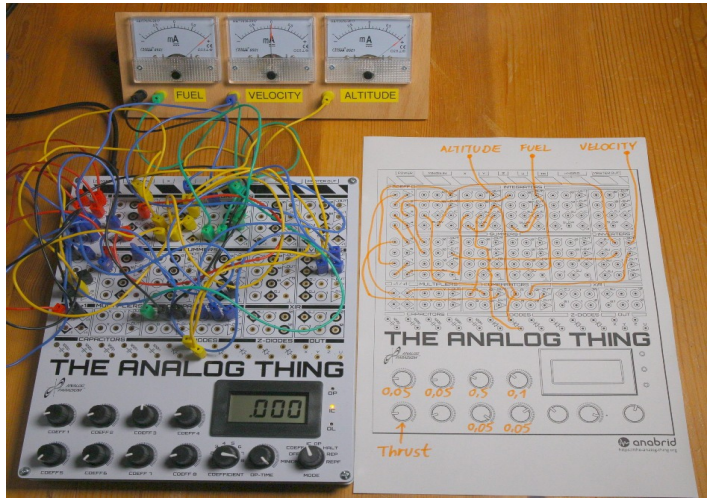
- **Bestimmung von Geschossflugbahnen mit dem Analogrechner**
  https://rclab.de/analogrechner/flugbahnen

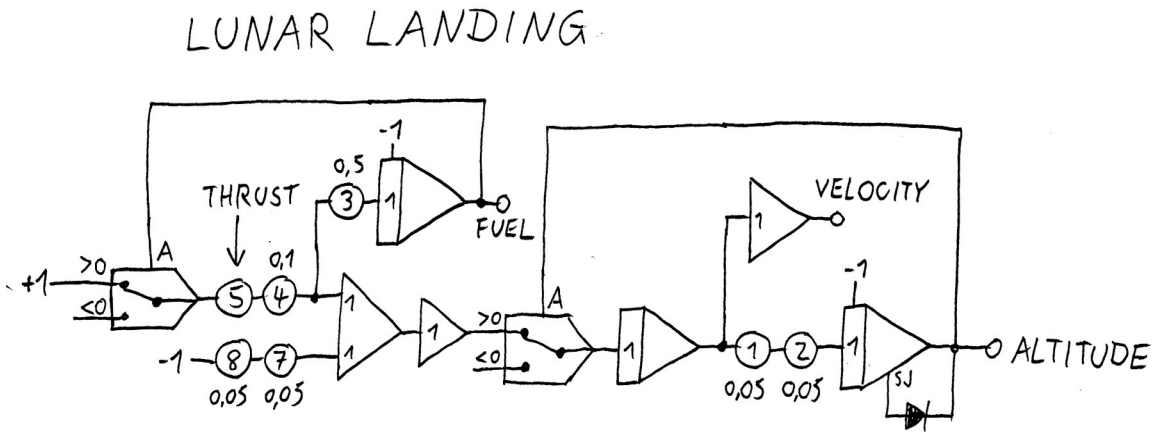
- **Growth of yeast:**
  http://www.kohorst-lemgo.de/modell/hefe/hefe.htm
  https://epub.ub.uni-greifswald.de/frontdoor/deliver/index/docId/758/file/diss_AlZoukra_Kristine.pdf

8 Examples without Scaling

8.1 Lunar Landing

An analog computer should have analog output devices. This is the lunar landing example from the manual, but I changed a few things. The coefficients are different, and I added a comparator at the input of the velocity integrator. If the height is negative, the input is switched to zero so that the touch-down velocity is saved. The goal of this game is to land the Eagle with the smallest possible touch-down velocity. Don't look out of the window. Look only on the instruments.



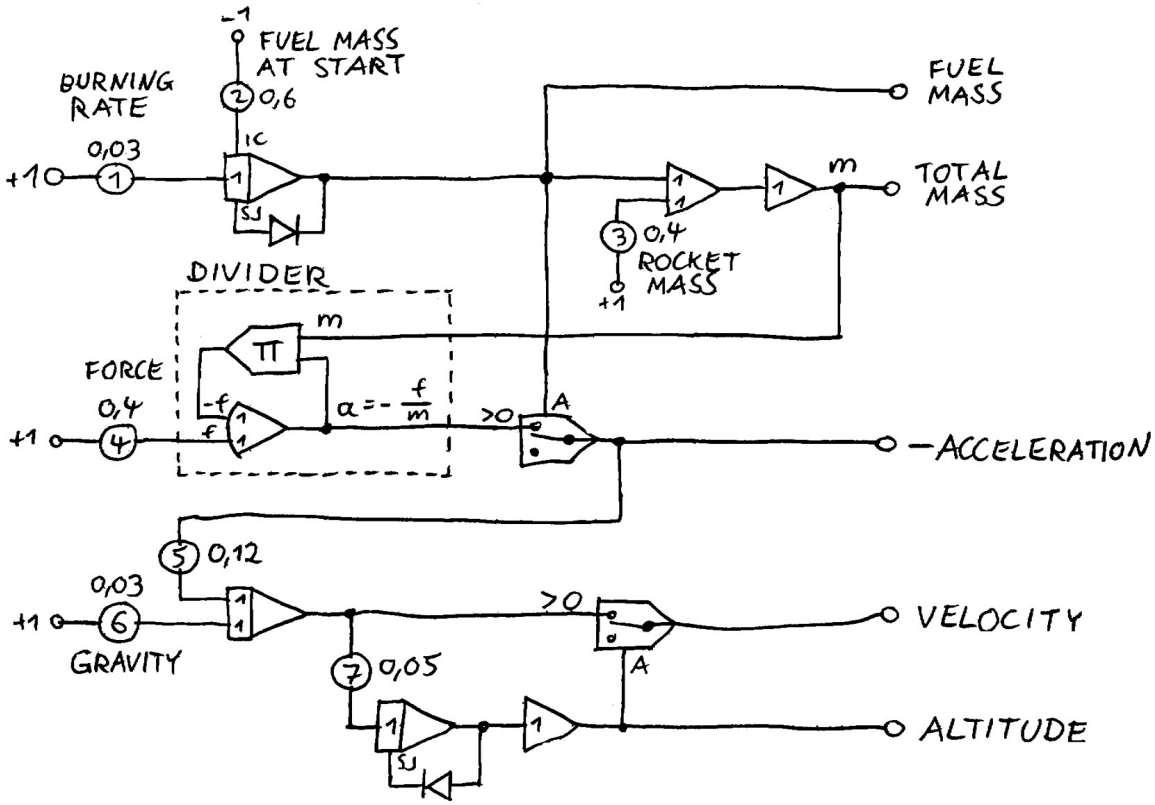


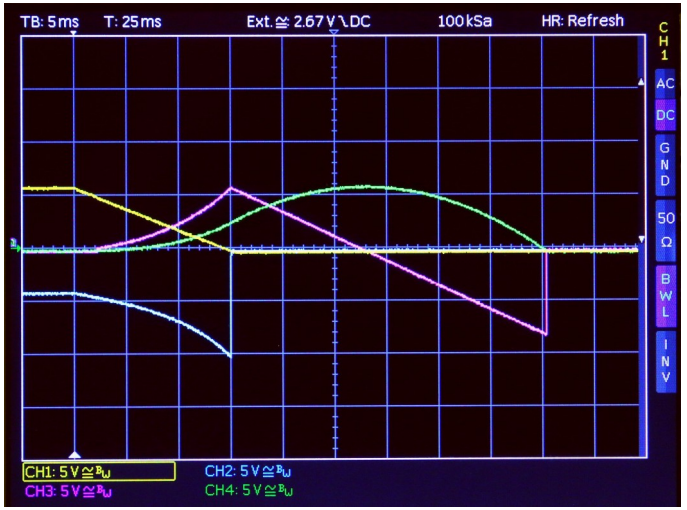
See also: https://www.youtube.com/watch?v=VNun_po0n6U

8.2 Rocket Start

This is a simulation of a rocket start, flight and crash landing. The total mass is the sum of the fuel mass (which gets less during the flight) and the mass of the empty rocket. Assuming that the motor has constant thrust, the acceleration increases because the total mass decreases. The acceleration reaches its maximum value just before the fuel tank is empty, and then drops to zero. For the rest of the flight the altitude over time is a parabola. Small error in schematic: The inverted acceleration is measured after the comparator, not before it.

Yellow = fuel mass Blue = acceleration (inverted) Red = vertical velocity Green = altitude

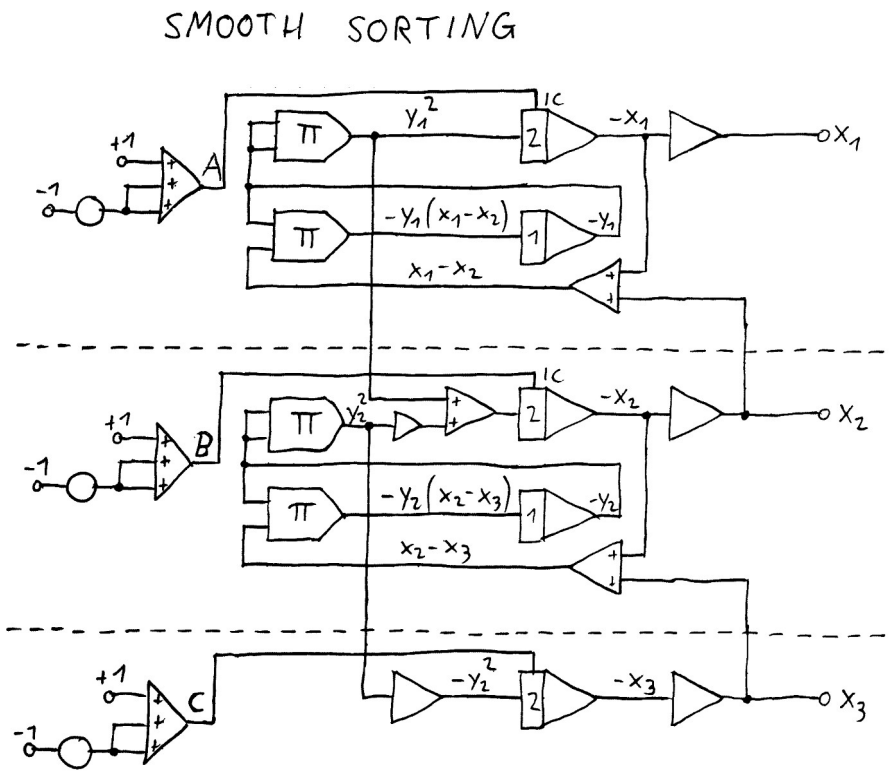


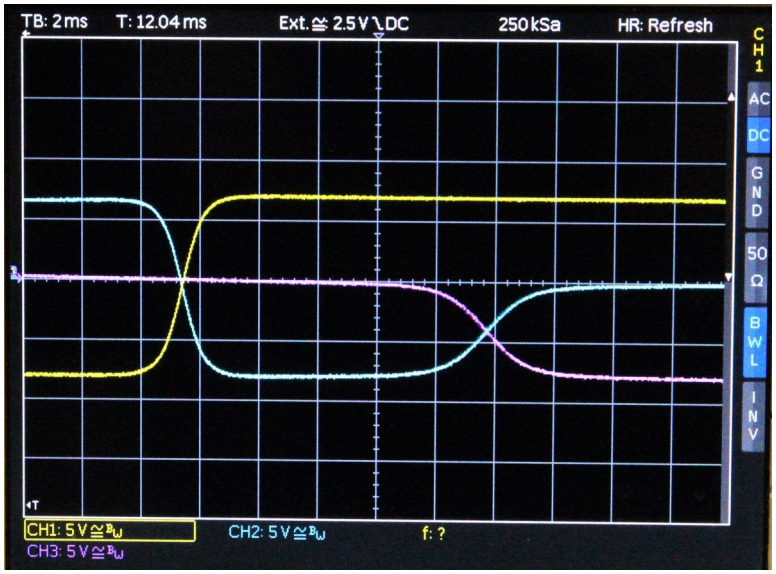


8.3 Smooth Sorting

This is an example for smooth sorting. Three input values A,B,C can be set to any values, and after some computing time will appear in sorted order at the three outputs. X1 is max(A,B,C) and X3 is min(A,B,C). This example with three inputs and three outputs runs on two THATs. If you need more inputs and outputs, you can duplicate the middle part (between the dashed lines). You can also remove the middle part, then you have only two inputs and two outputs and it will fit on one THAT.

The equations are from this paper: http://www.hrl.harvard.edu/analog/

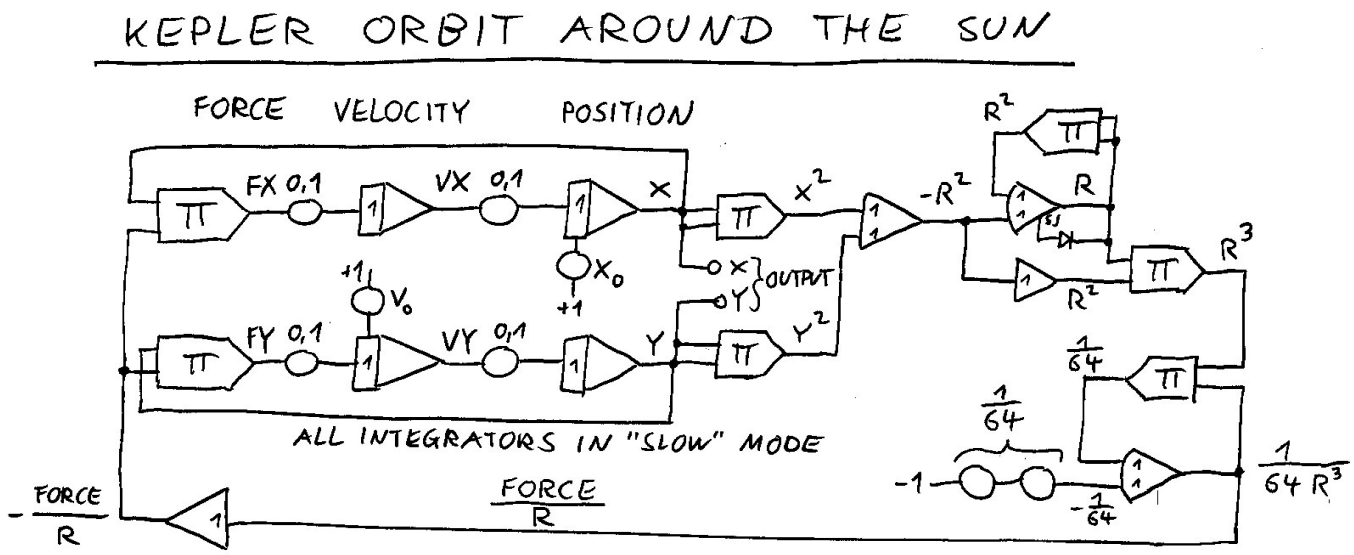




8.4 Kepler Orbits

This is a simulation of Kepler orbits (circle, ellipse, parabola, hyperbola). Unfortunately 7 multipliers are required, which means 4 THATs. I would really like to save at least one of the multipliers, but I have no idea how that could be done. (Update: Using the Teensy LC board with a look-up-table with $-R^2$ as input and $1/(64\ R^3)$ as output will save three multipliers, so that it will run on two THATs).
The sun is in the center at x=0, y=0. It's important that the distance R between the planet and the sun is always in the 0.25 to 1 range. If it's larger, then R does overflow, and if it's smaller then 1/(64·R^3) does overflow.

See also: https://analogparadigm.com/downloads/alpaca_11.pdf
See also: https://www.facebook.com/groups/theanalogthing/posts/950856693803787/

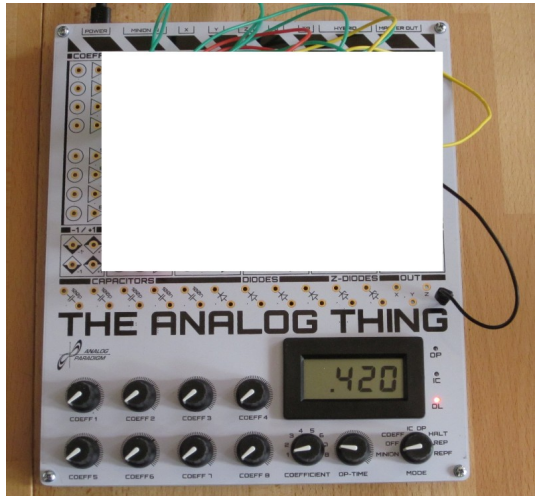


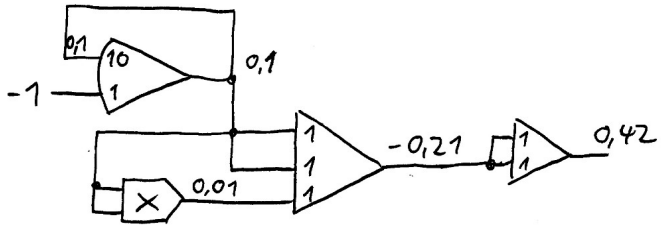
R	R^2	R^3	$1/R^3$
0,25	0,062	0,0156	64
0,5	0,25	0,125	8
0,75	0,56	0,42	2,37
1	1	1	1

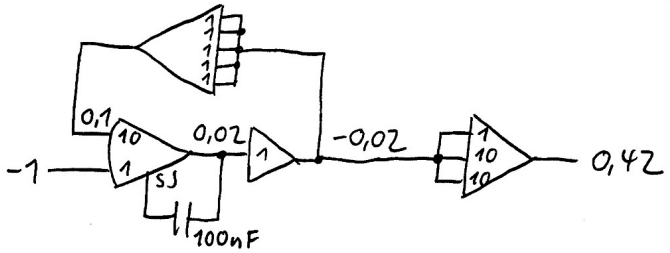
$0{,}25 < R < 1$

8.5 Calculate 0.42

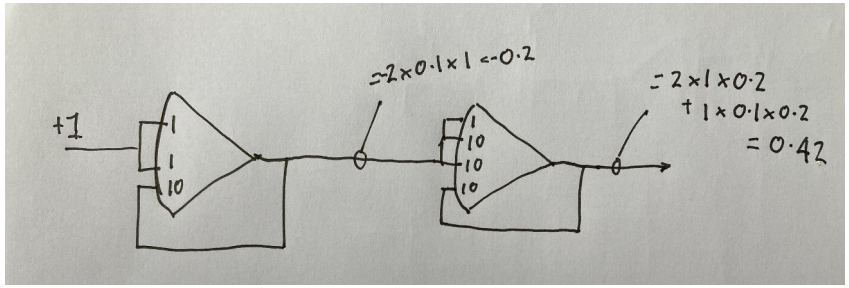
Design a circuit that calculates the answer to everything (42) normalized to 100. You are not allowed to use any coefficient potentiometers or external circuits. It must run on one THAT. Of course, there may be more than one correct solution.







This solution is from Chris Giles:

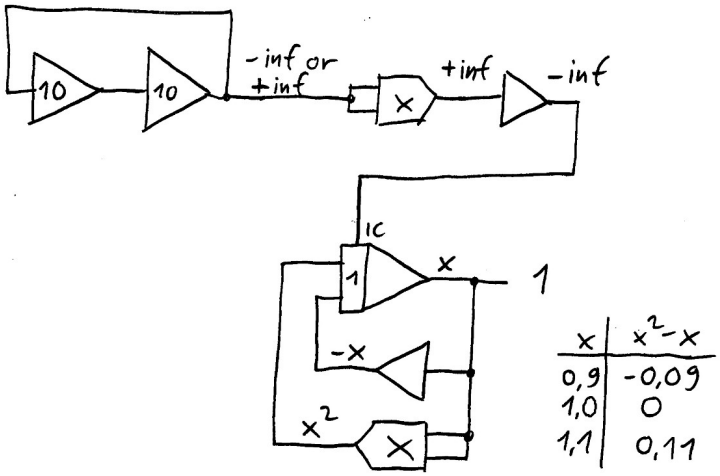


8.6 Calculate the machine unit 1

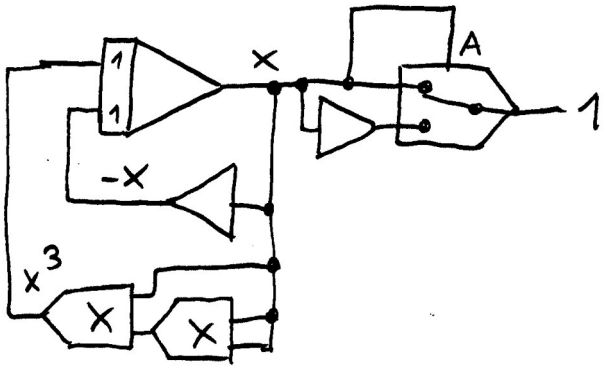
Design a circuit that calculates the +1 machine unit, without using the -1 or +1 signals. You are not allowed to use coefficient potentiometers or external components. It must run on one THAT. Small rounding errors in the last digit are no problem.

This is one possible solution. Let's first look at the lower half of the circuit. It's an integrator with x^2 - x in the feedback loop. Unfortunately this circuit has two stable results: -infinity or +1. To which state it runs depends on the sign of the initial condition. With the default zero initial condition, it may run either to -infinity or to +1. That means we must connect a negative voltage to the IC input. But it's not allowed to use the -1 signal. How can a negative signal be created from nothing? Two summers are connected in a loop with gain 100. It's almost impossible that the output voltage stays at 0 for more than a millisecond. After a very short time the output must be either -infinity or +infinity. We don't know which one, because that depends on the offset errors of the Op-Amps. But after squaring the signal, we know that the result must be +infinity, and after inverting we have -infinity.

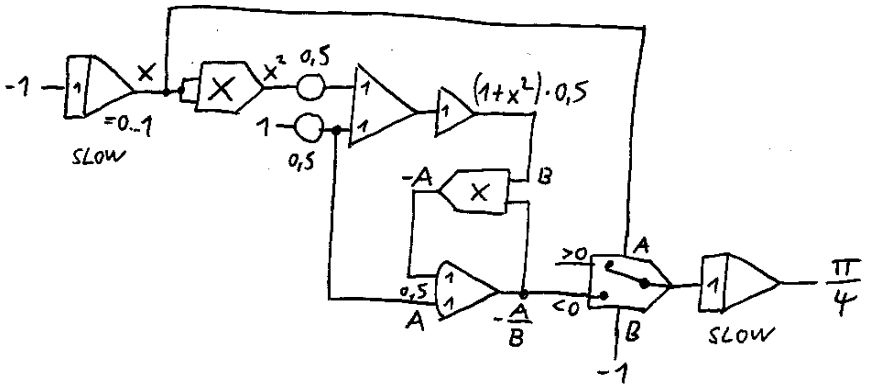
By the way, the equation x^2 - x = 0 does also have the solution x=0, but x=0 is instable in this circuit.



This solution is better, because it doesn't need an initial condition. The feedback of the integrator is x^3 - x, which means the integrator is running to -1 or +1. A comparator inverts the output, if negative.

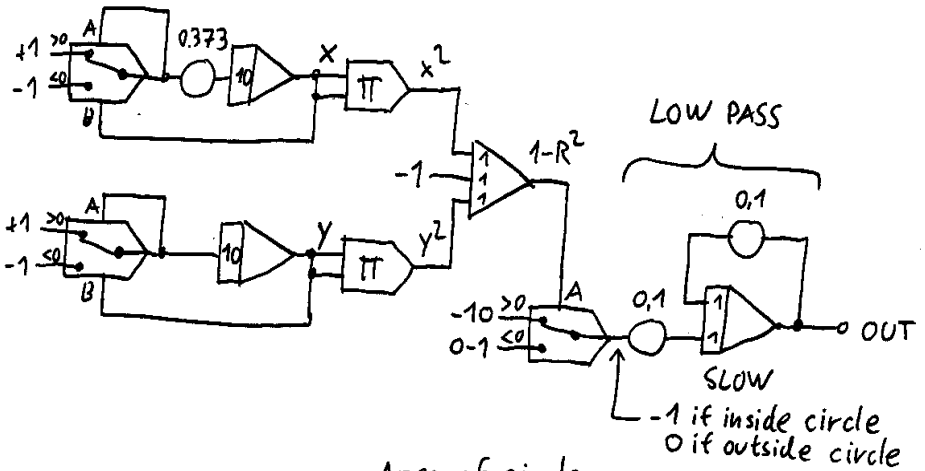


8.7 Calculate π/4



Calculate π/4

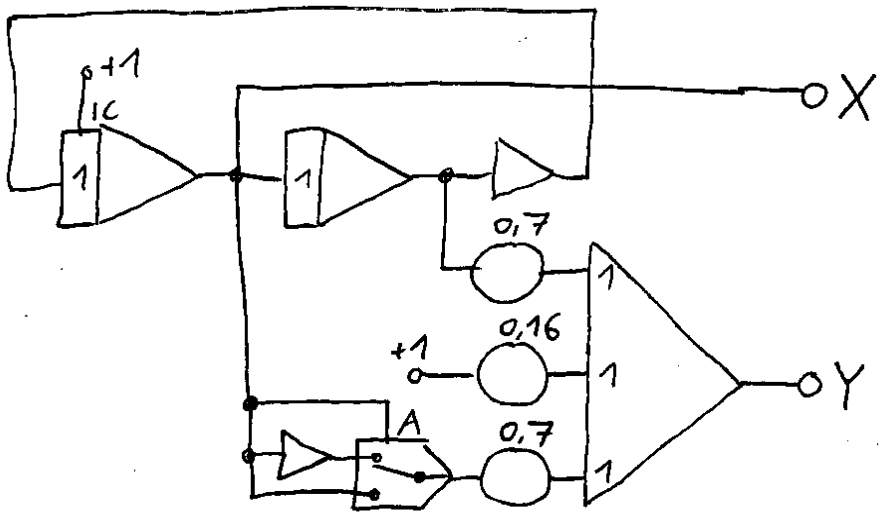
$$\frac{\pi}{4} = \int_0^1 \frac{dx}{1+x^2} \approx 0{,}785$$

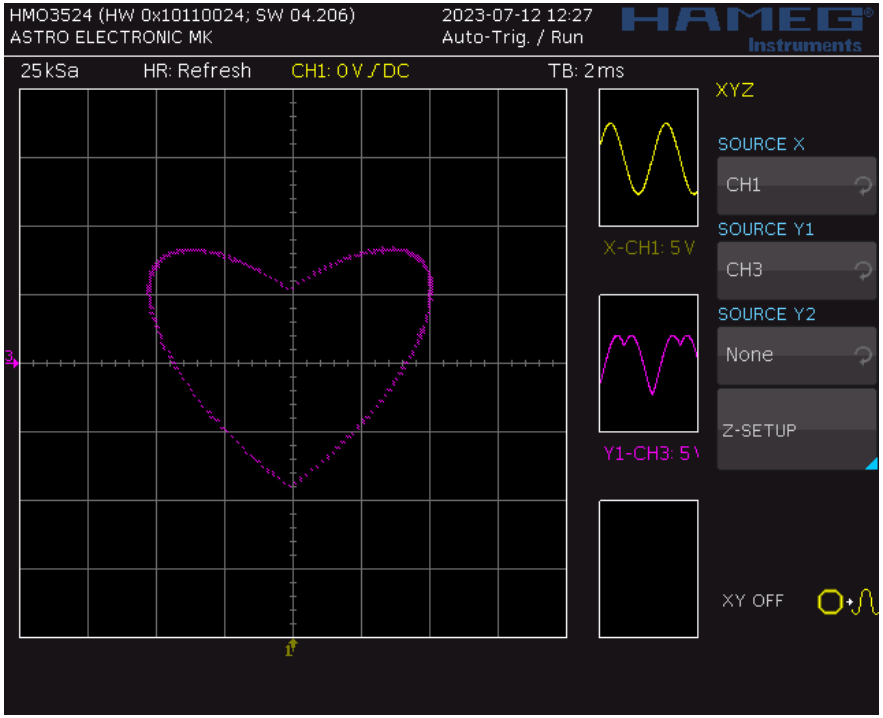


Calculate $OUT = \frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi}{4}$

Problem: The comparators in the triangle wave generators are too slow, so that the amplitude in slightly larger than ±1

8.8 Heart Curve



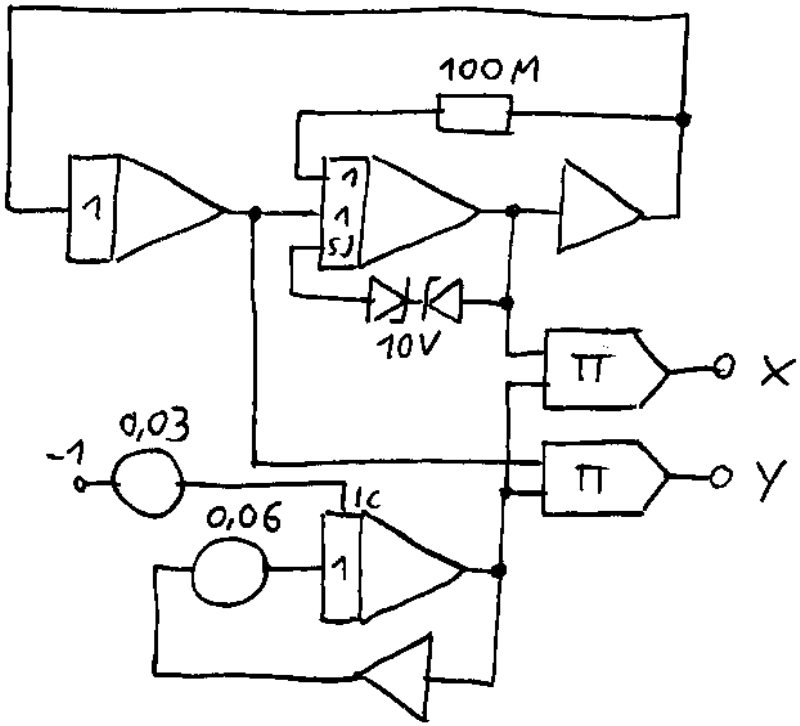


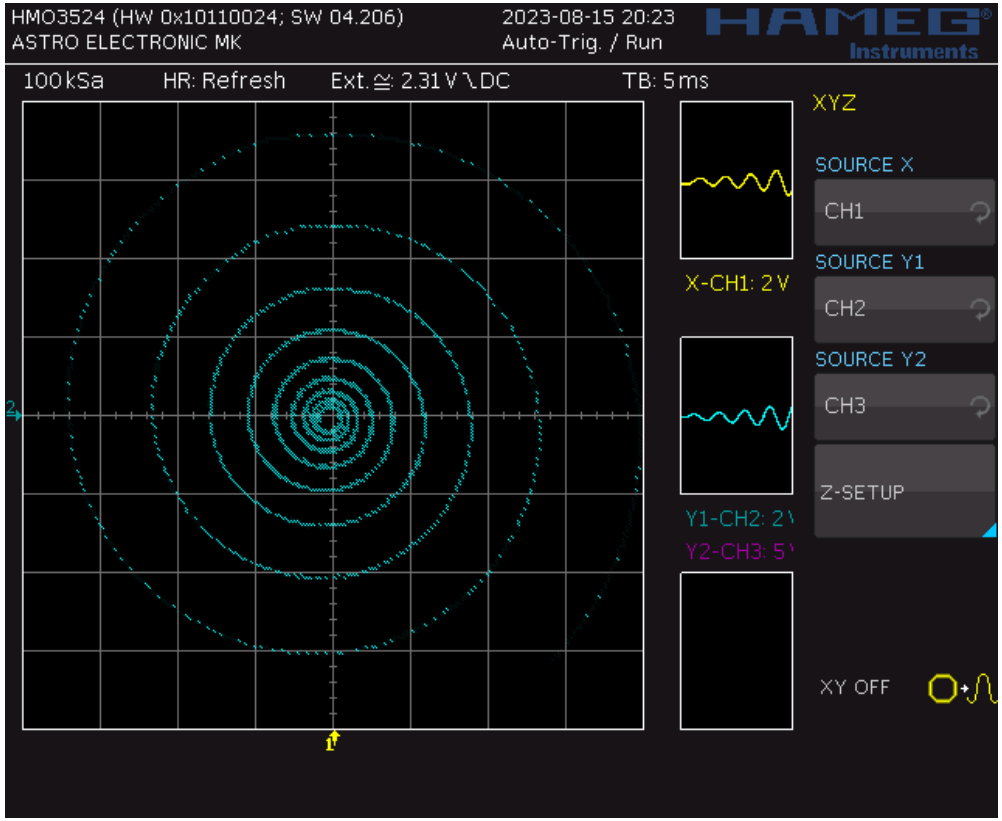
See also:

Hamid Naderi Yeganeh: „Making Mathematical Art“ https://www.scientificamerican.com/blog/guest-blog/making-mathematical-art/
Hamid Naderi Yeganeh: „How to Draw with Math“ https://www.scientificamerican.com/blog/guest-blog/how-to-draw-with-math/

https://www.facebook.com/HamidNaderiYeganeh

8.9 Logarithmic Spiral

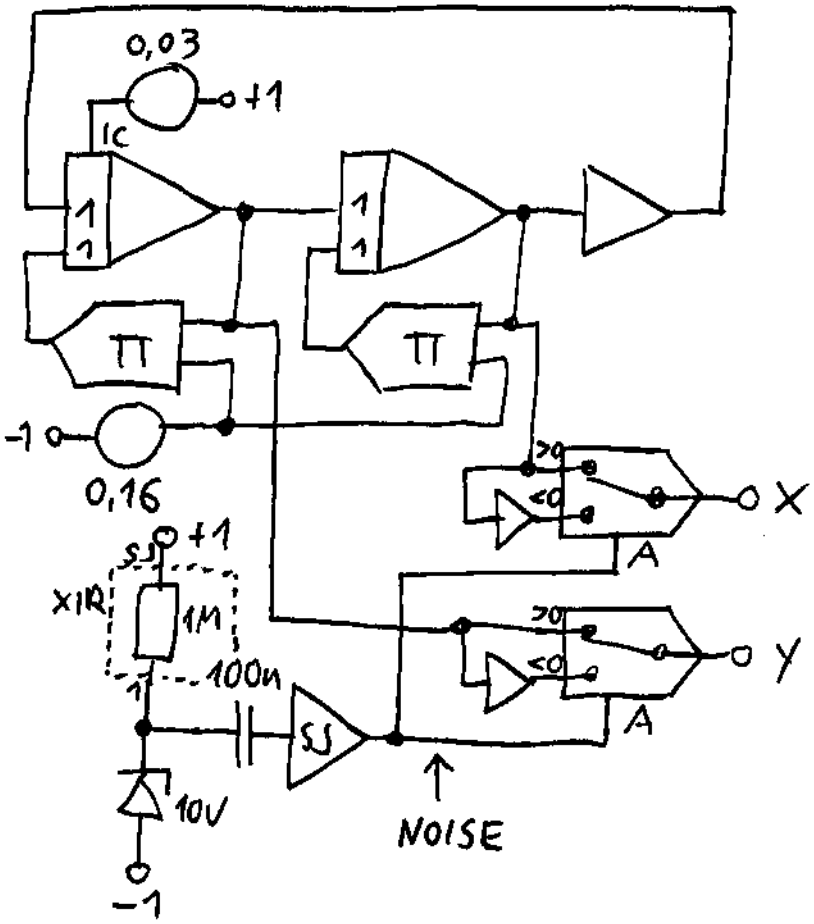


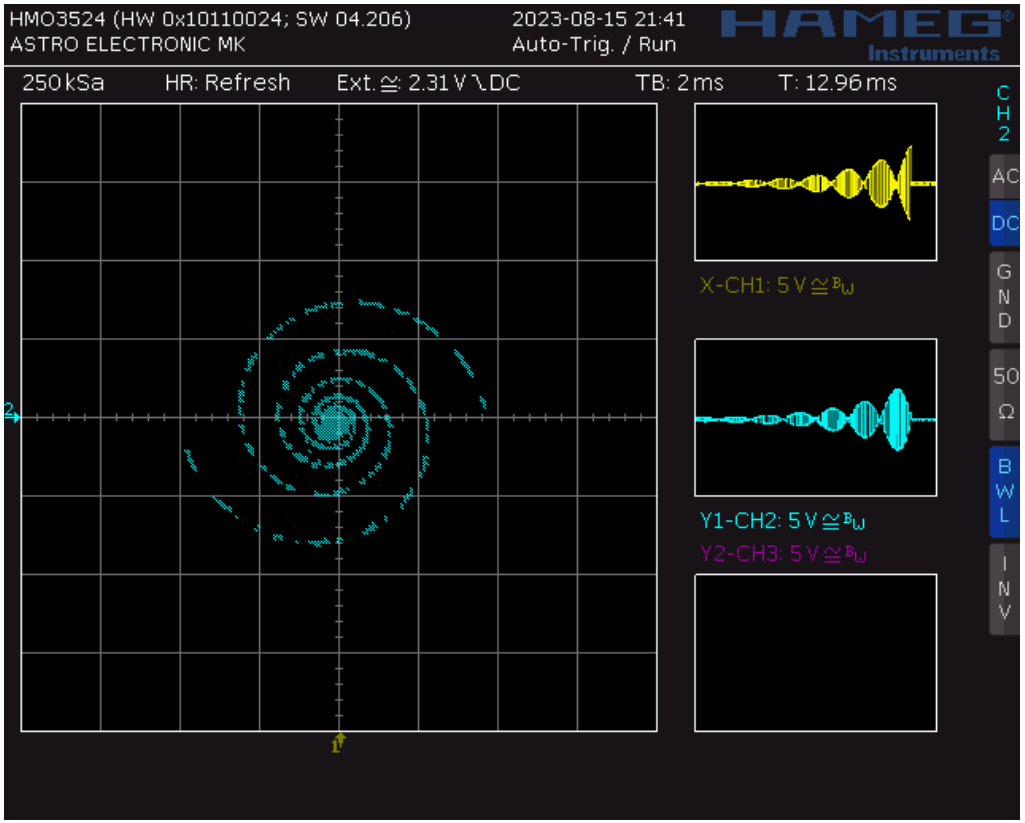


See also: https://en.wikipedia.org/wiki/Logarithmic_spiral

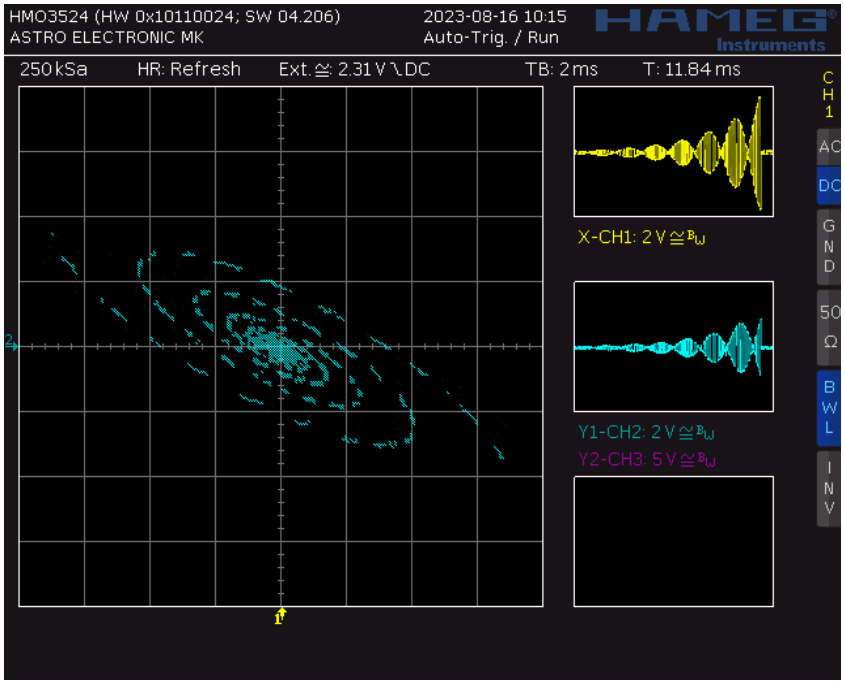
8.10 Spiral Galaxy Generator

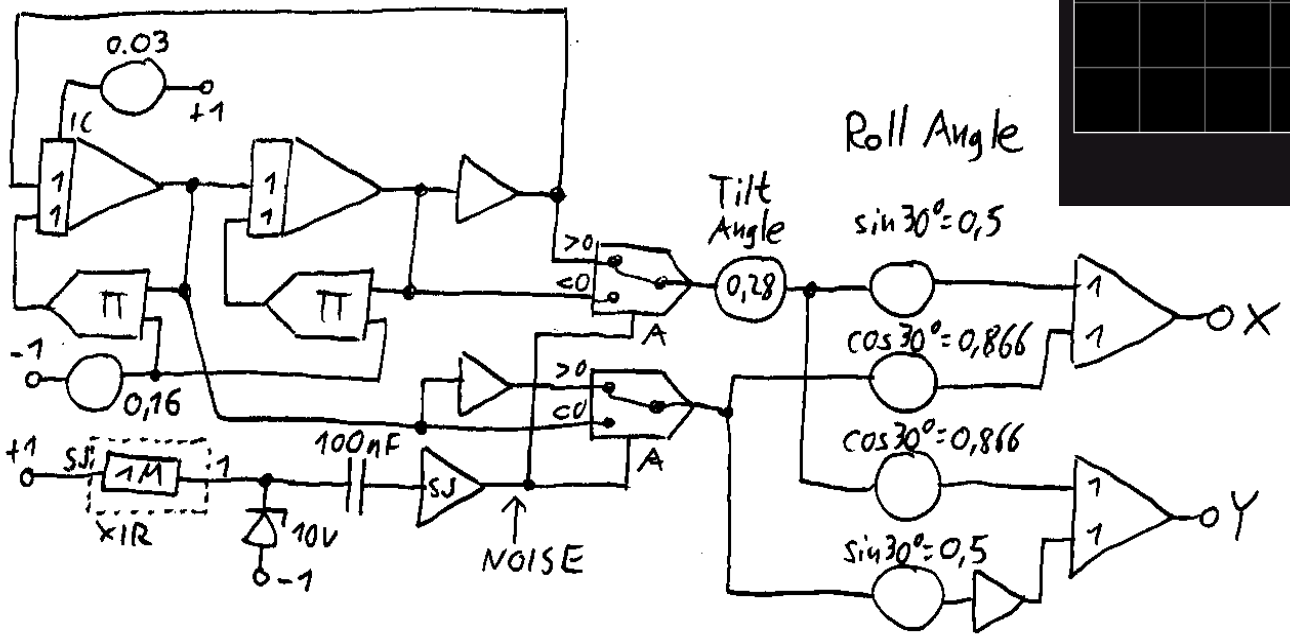
These are two logarithmic spirals, multiplexed by a noise signal.





Let's add a rotation matrix. With the "Tilt Angle" coefficient you can select between edge-on and face-on galaxies:





8.11 Levitating Magnet

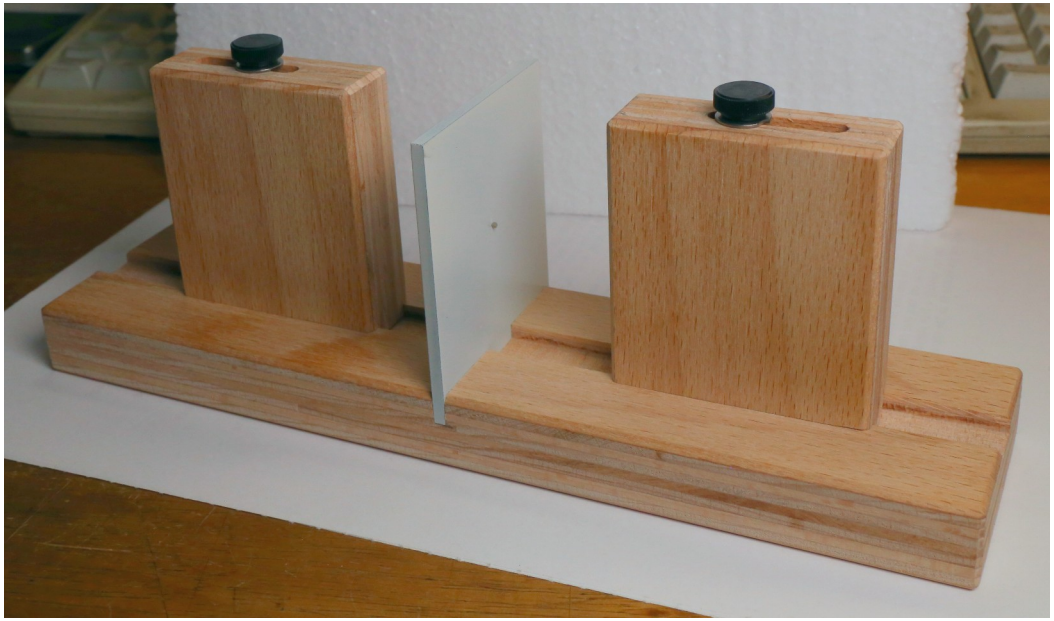
The theoretical background: Earnshaw's theorem states that there can be no arrangement of permanent magnets that allows a small permanent magnet to float stably and freely in space. All conceivable arrangements are unstable. It is only possible if an active control loop is used.

Earnshaw's theorem: https://en.wikipedia.org/wiki/Earnshaw%27s_theorem

This device apparently shows that it's indeed possible to let the magnet float stably in space. There is no active control built in. No current, no electronics, no superconductivity, no low temperature.

This is not my idea. I've only built the device.

The small 1mm^3 cube magnet is barely visible in the center of the white plate:



How does it work?

Two strong permanent magnets are hidden in each of the two wooden blocks. They are cube-shaped with an edge length of 10 mm. The small free-floating magnet has an edge length of 1mm. All three magnets are oriented in such a way that they attract each other. This means that all north poles point to the left and all south poles point to the right. In this arrangement, there is a point exactly in the middle where the attractive forces of the two large magnets on the small magnet compensate each other. This point is stable when the small magnet moves in the plane of the plate. When it is deflected, it is always drawn back to the center of the plate. But this point is unstable when it comes to left/right movement. This means that if the small magnet moves a tiny bit to the left or right (i.e. in the direction of one of the two large magnets), the attractive force of this magnet prevails and the small magnet flies towards this magnet.

Now the plate comes into play. It is made of graphite, which is a diamagnetic material. The small magnet is repelled by the plate, but this force is very small and only acts over a short distance.
The small magnet is not exactly in the middle between the two magnets, but a little to the left of the center. It would therefore fly to the left magnet if this were not prevented by the repulsive force of the plate.
The stable area for the small magnet is very small in the left/right direction, about +-0.2mm. The device only works if the distances are set very precisely. If the small magnet moves too far away from the plate, the attractive force of the right magnet will prevail and it will fly there.

I found the idea in this video (in german): https://www.spektrum.de/.../levitation-bei.../2171757

Does the device violate Earnshaw's theorem? No, because diamagnetic materials aren't allowed in Earnshaw's theorem.

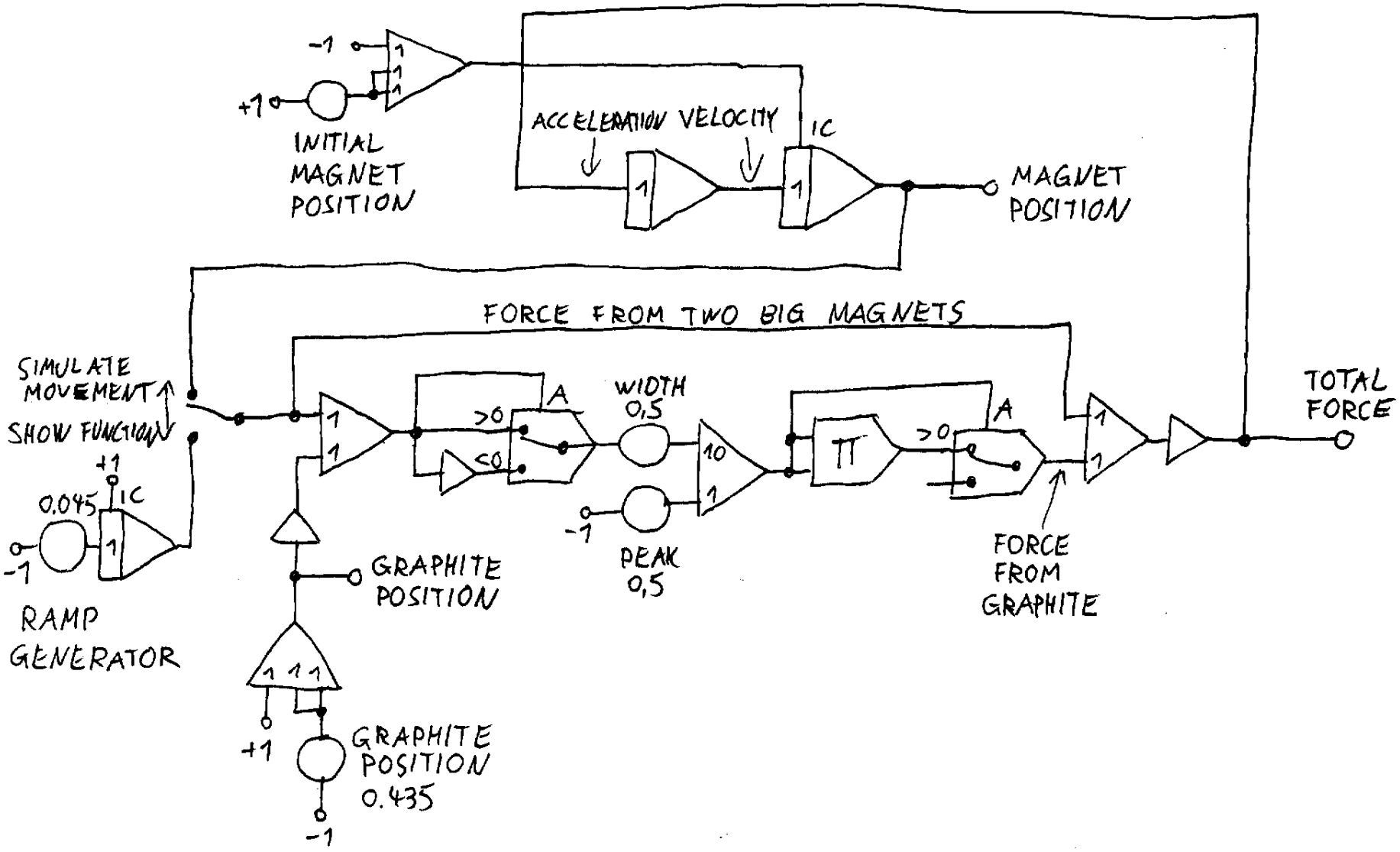
Now let's try to simulate this effect on the analog computer.

The circuit has two integrators for the small magnet's velocity and position.
Position is definded as -1 for the left big magnet and +1 for the right big magnet.
The position is fed into a function which calculates the force which acts on the small magnet.

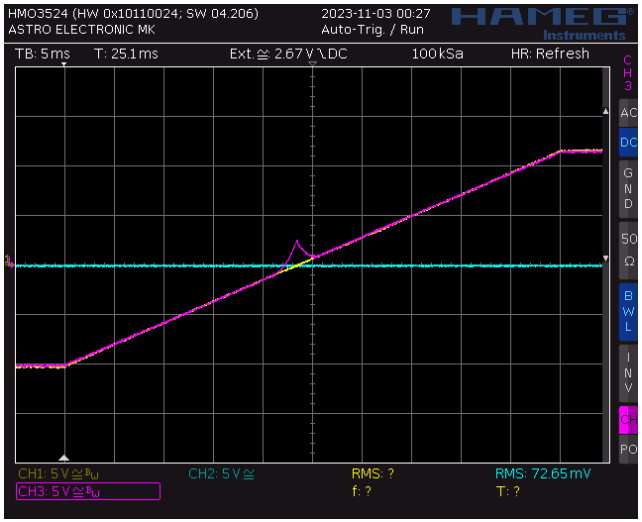
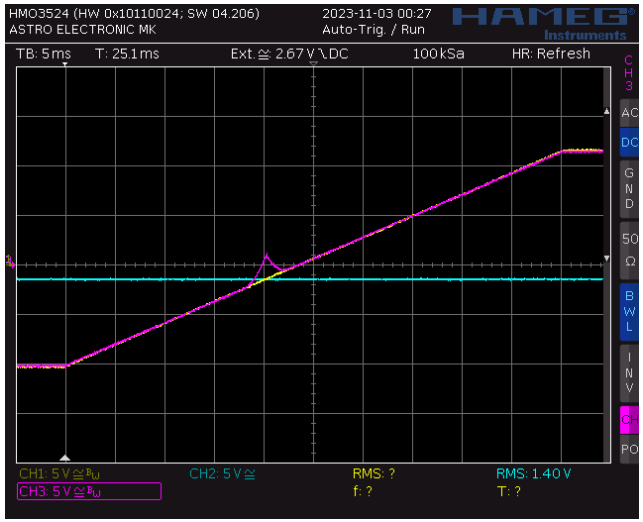
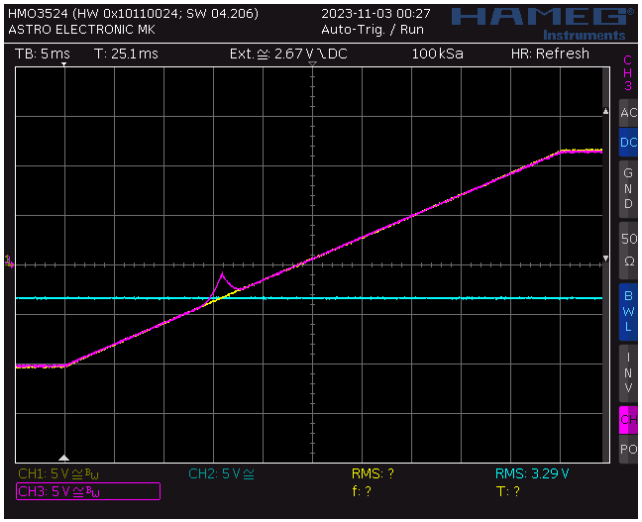
The small magnet's position can only be stable if this function crosses the zero line with a negative slope.
Without the graphite plate, there is no stable position because the slope is always positive.
A stable position is only possible near the graphite plate, if the magnet is a little bit left of the center, and very close to the plate.

The circuit has a switch where you can select two modes:

- **In upper position the movement of the small magnet is simulated. Use "Magnet Position" as output.**
- **In lower position a ramp generator is connected to the input of the function, so that you can see how the function looks like. Use "Total Force" as output.**


-1
+1
INITIAL MAGNET POSITION
ACCELERATION VELOCITY
IC
MAGNET POSITION
FORCE FROM TWO BIG MAGNETS
SIMULATE MOVEMENT
SHOW FUNCTION
+1
0.045
IC
-1
RAMP GENERATOR
>0
<0
A
WIDTH 0,5
10
-1
PEAK 0,5
GRAPHITE POSITION
+1
GRAPHITE POSITION 0.435
-1
TT
>0
A
FORCE FROM GRAPHITE
TOTAL FORCE

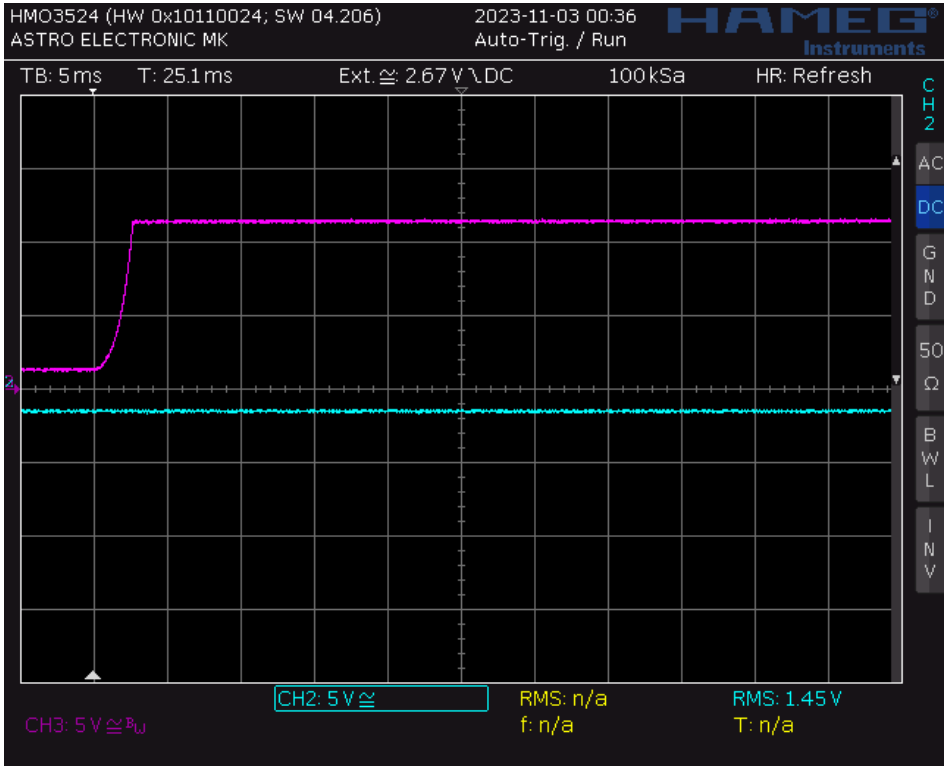
Here are a few examples how the function looks like:

The graphite plate (cyan) is exactly in the center between the two big magnets. The total force (magenta) has a small region with negative slope, but this region doesn't cross the zero line. There is no stable position for the small magnet.	The graphite plate (cyan) is moved a little bit to the left (negative). The total force (magenta) has a small region with negative slope, and this region does cross the zero line. This small region is a stable position for the small magnet.	The graphite plate (cyan) is moved even more to the left (negative). The total force (magenta) has a small region with negative slope, but this region doesn't cross the zero line. There is no stable position for the small magnet.
		

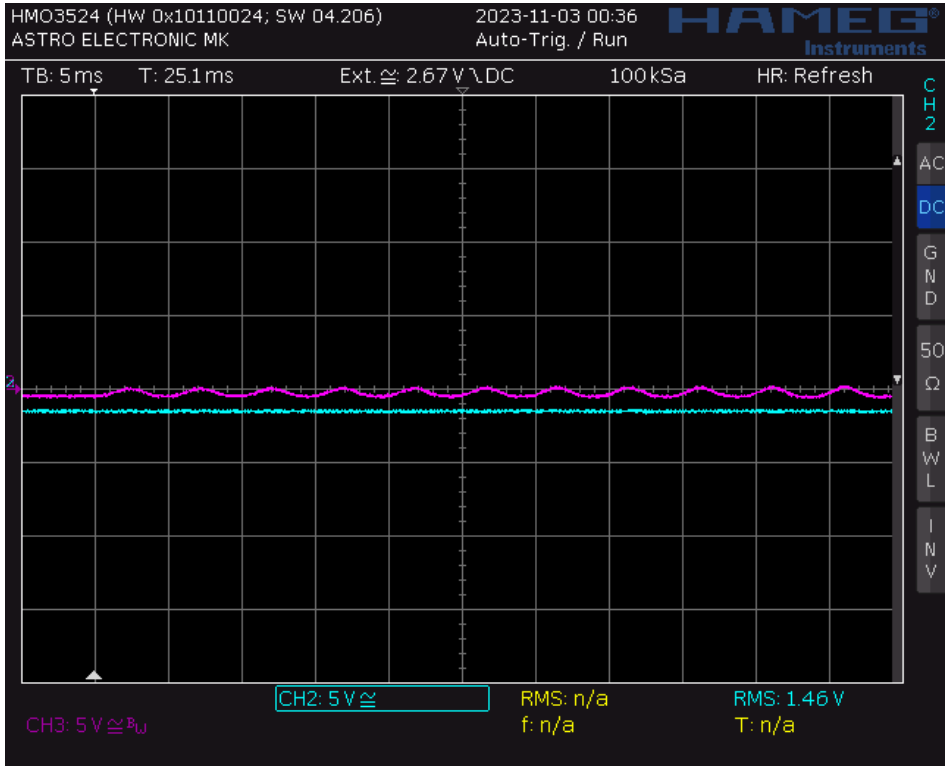
By the way, the circuit is very similar to a tunnel diode oscillator.

Here are two examples for the simulation of the small manget's position over time. In both examples the function (and the graphite plate position) is set as in the middle example on the previous page.

Here the initial position of the small magnet (magenta) is too far away from the graphite plate (cyan), so that the magnet does immediately fly to the right big magnet (positive).



Here the initial position of the small magnet (magenta) is close enough to the graphite plate (cyan), so that the magnet does oscillate around a stable position near the graphite plate.



8.12 Tunnel Diode Oscillator

https://en.wikipedia.org/wiki/Tunnel_diode

AE.1.1 Vergleichsmerkmale von Germanium-Tunneldioden

Typ / Kennfarbe Kennzeichen	Her-steller	Bild	Gehäuse Mat	Gehäuse Norm.	zus. Kühlung KS KK	Kühlblech Mat	Kühlblech Maße mm cm²	Techno-logie	f_g MHz	f_r MHz	$-r_n$ Ω	$-r_{nstr}$ Ω	r_o Ω	r_{ostr} Ω	l_o nH	c_o * c_G pF	c_{ostr} pF	I_p mA	I_{pstr} mA	U_p mV	I_v mA	U_v mV	$\frac{I_p}{I_v}$	U_{Fp} bei I_p mV	I_{FM} * I_{RM} # i_{FM} mA	P_{tot} mW	T_{jM} * T_{UM} # T_{Umin} °C
			mechanisch-technologische						elektrische (Kenn- und Grenzwerte bei T_U = 25 °C)																		
TU 4 bl	S	26	K					lg			60	30...100	1,5	<2	1,2	20 *0,5	10...30	1,6	1,3...2	55		250	7		3 *3		*100 #-50
TU 5 ws	S	26	K					lg			90	60...150	2	<3	1,2	10 *0,5	5...20	1,3	0,8...1,6	55		250	7		3 *3		*100 #-50
AE 100	T	69	K					lg			100		1		5	10		1		55	0,15	300				20	
AE 101	T	69	K					lg			100		1		0,5	10		1		55	0,15	300				20	
TU 6 li	S	26	K					lg			130	80...200	3	<6	1,2	5 *0,5	2...10	0,8	0,7...1,2	55		250	7		3 *3		*100 #-50
TU 7 sw	S	26	K					lg			130	80...200	2	<4	1,2	7 *0,5	4...10	1	0,85...1,15	55		250	7		3 *3		*100 #-50
TU 8 li	S	78	K					lg			130	80...200	3	<6	0,75	5 *0,5	2...10	0,8	0,7...1,2	55		250	7		#3		*100 #-50
TU 9 sw	S	78	K					lg			130	80...200	2	<4	0,75	7 *0,5	4...10	1	0,85...1,15	55		250	7		#3		*100 #-50
TU 2 ge	S	26	K					lg			150	<250	1,5	<2	1,2	30 *0,5	<50	1	0,6...1,4	55		250	7		3 *3		*100 #-50
TU 3 rt	S	26	K					lg			150	<250	2	<3	1,2	15 *0,5	<25	0,5	0,25...0,75	55		250	7		3 *3		*100 #-50
AEY 12	SEL	30	M	TO-18				lg	980	200	20		0,4		6	90		15	13,5...16,5	55	1,8	310	8	490	150 *150		75
AEY 11	SEL	30	M	TO-18				lg	1050	310	25		1		6	30		5	4,5...5,5	55	0,7	300	7	480	50 *50		75
JK 30A	SEL	55	K					lg	1100	900	25		0,9		1	30		5	4,5...5,5	60	0,7	320	7	480	50 *50		75
JK 60A	SEL	56	K		=2xJK 30A, Paar																						

Beispiele von Tunneldioden

Tabelle II

Typ	Art	Hersteller	Strom und Spannung im Höcker		Strom und Spannung im Tal		Höcker/Tal Stromverhältn.	Negativer Widerstand	Kapazität	Eigeninduktivität
			mA	mV	mA	mV		Ω	pF	nH
1N2939A	Ge	General Electric	1	60	0,1	350	10	150	4...5	6
1N3150	Ge	General Electric	22	60	0,9	350	8	10	60	6
TU10/1	Ge	Siemens	1	55	0,14	250	7	110	2	0,75
TU14/1	Ge	Siemens	20	55	3	250	7	10	30	1,5
JK19A	Ge	St. El. & C.	1	55	0,16	290	6	110	10	8
JK21A	Ge	St. El. & C.	15	55	2,5	310	6	8	80	8
AE100	Ge	Telefunken	1	55	0,15	300	7	100	10	5
A650	Ga- -As	Texas Instruments	10		0,67		15		25	

More info about tunnel diodes: https://w140.com/tekwiki/wiki/Tunnel_diodes

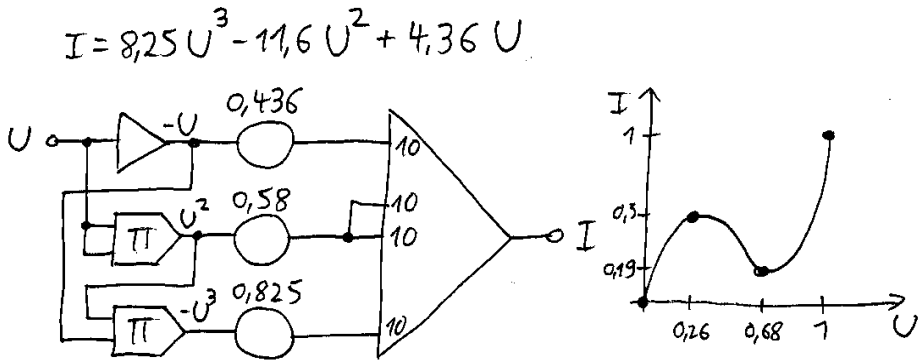
Russian tunnel diodes: https://w140.com/tekwiki/wiki/Russian_tunnel_diodes

AI201G: https://www.richis-lab.de/Diode07.htm

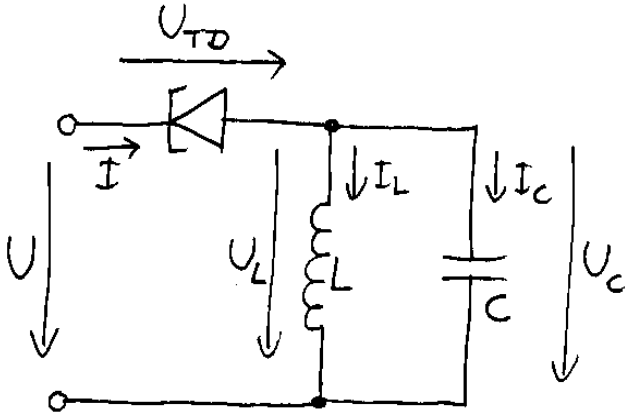
Tunnel diodes from American Microsemiconductor: https://web.archive.org/web/20211016024914/https://www.americanmicrosemi.com/tutorial/tunnel-diode-and-back-diode/

Tunnel diode pulser: https://www.amplifier.cd/Test_Equipment/Tektronix/Tektronix_other/Tunnel_Diode-Pulser/Tunnel_Diode_Pulser.html

Current over voltage characteristic of tunnel diode (this is a very rough approximation):



Tunnel diode oscillator with equations:



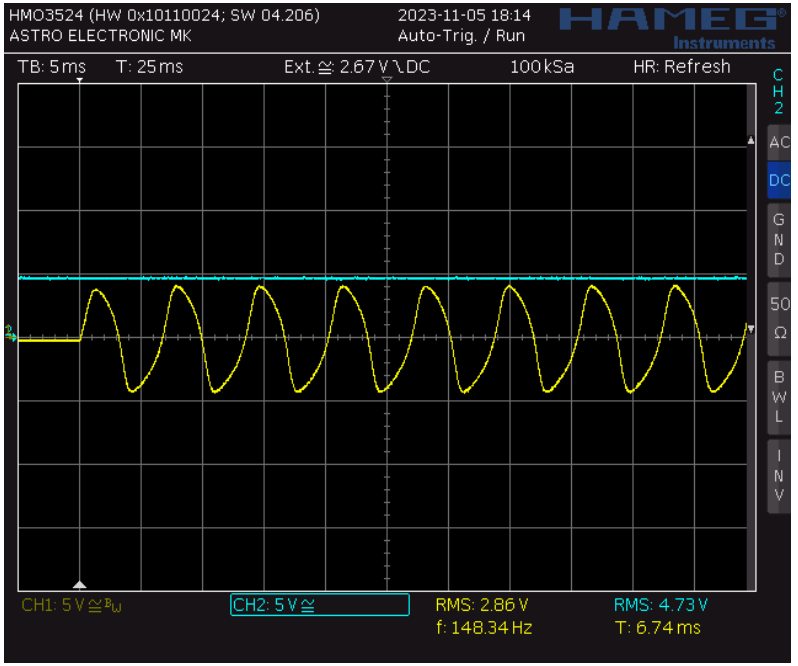
$$\frac{dU_C}{dt} = \frac{I_C}{C}$$

$$\frac{dI_L}{dt} = \frac{U_L}{L}$$

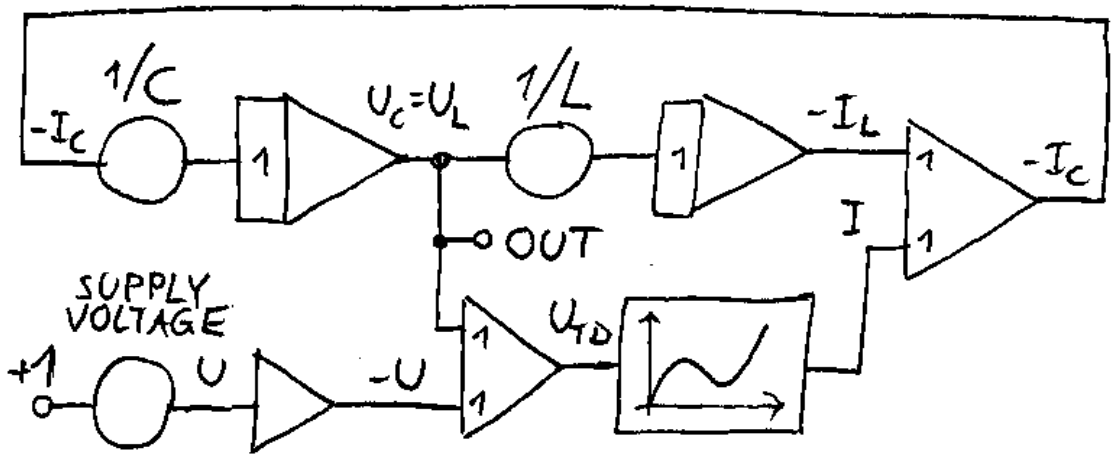
$$I = I_L + I_C$$

$$U = U_{TD} + U_C$$

$$U_C = U_L$$

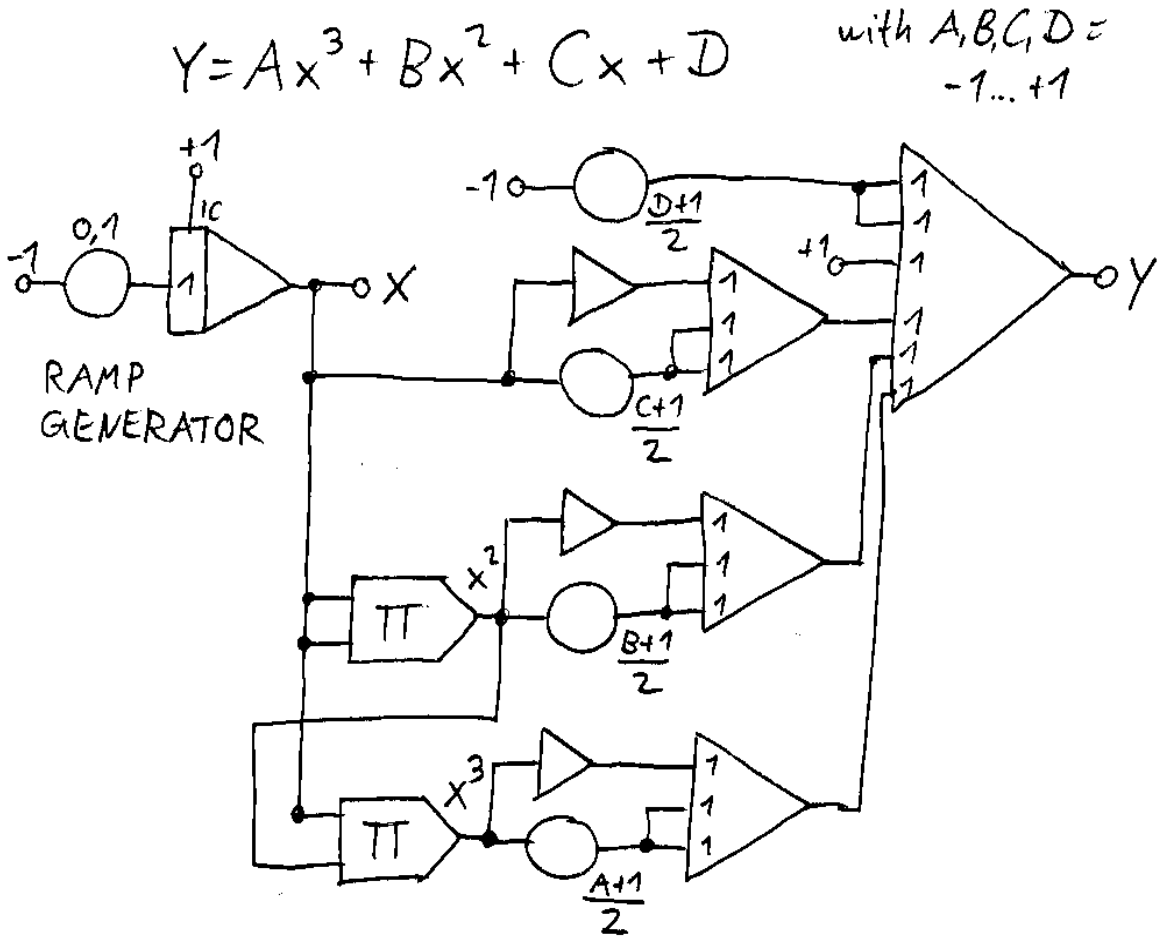


Circuit for tunnel diode oscillator:



8.13 Polynomial Generator y = ax^3 + bx^2 + cx + d

The parameters A, B, C and D are adjustable from -1 to +1.



8.14 Tritone Generator

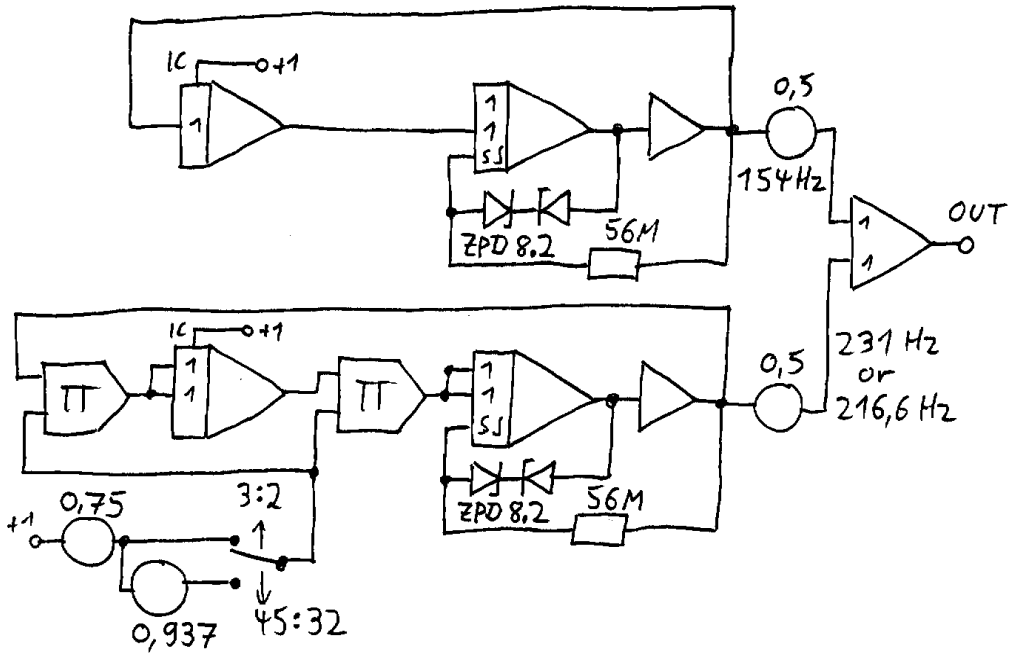
I found it in this book: Michael S. Horn, Melanie West, Cameron Roberts: Introduction to Digital Music with Python Programming, page 62

A tritone in music is defined as a 45:32 interval (or $\sqrt{2}$:1 interval) which was frequently avoided in composition for its dissonant qualities.

See also: https://en.wikipedia.org/wiki/Tritone

However a 3:2 interval sounds harmonic and is called a "Perfect Fifth". Which means it is the interval between the base note and the fifth note of a diatonic scale on that base note, and it is perfect because it is a 'pure' 3/2 ratio, not the 'impure' $2^{(7/12)}$ = 1.498 interval you find on a piano.

This circuit simulates both intervals, selectable by a switch. Tune the 0.75 coefficient for exact 3:2 ratio, which can easily be checked with an oscilloscope. The other coefficient is (45/32) / (3/2) = 0.9375. Connect the output to an audio amplifier.
For the exact $\sqrt{2}$:1 interval set the second coefficient to 0.943.



8.15 Guitar String Simulator

Giloi, Wolfgang / Lauber, Rudolf: Analogrechnen, page 259 - 264

Adler, Helmut: Elektronische Analogrechner (from 1962-1970), page 304 - 308

String vibration, english: https://en.wikipedia.org/wiki/String_vibration

german (much more details): https://de.wikipedia.org/w/index.php?title=Saitenschwingung

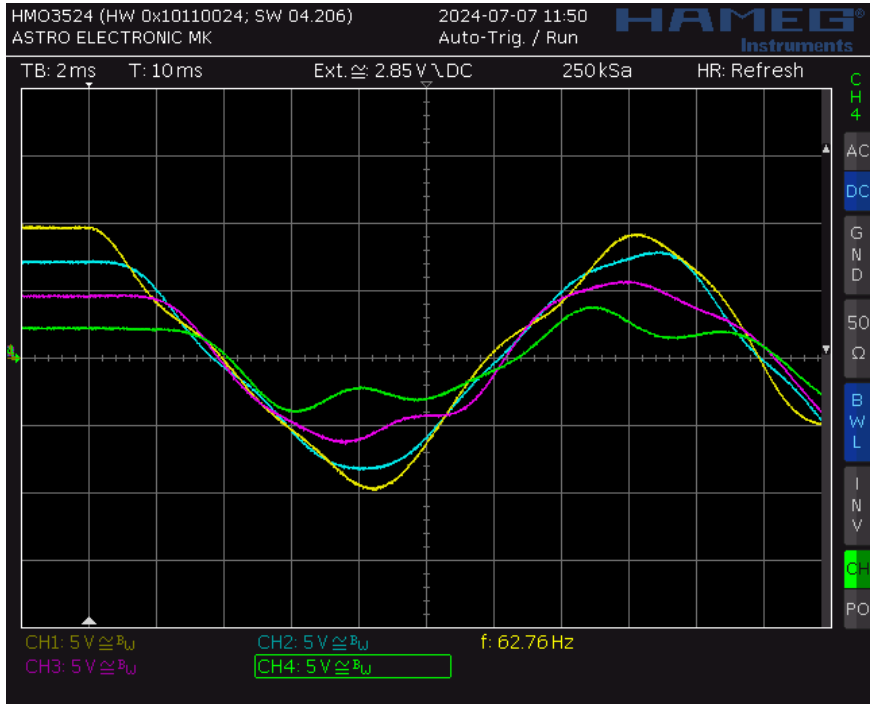
For simulation of guitar strings you need many integrators, two for each point on the string. The result becomes more accurate, if more points are calculated.

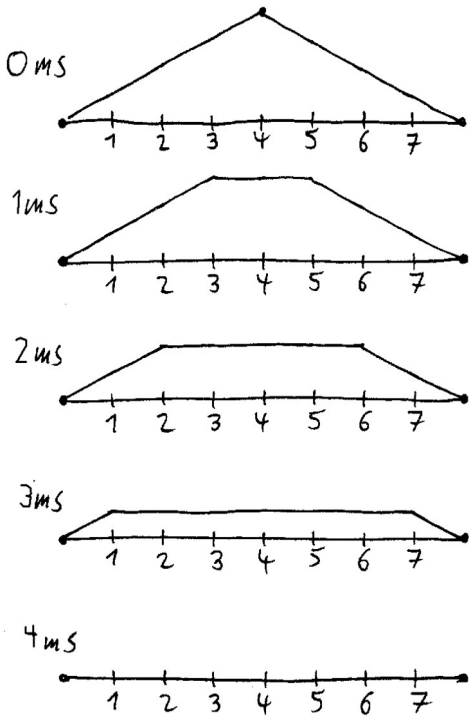
String is divided into n segments of equal length	Frequency with all coefficients = 1 Integrator TC=1ms	General case: Full string is simulated, no symmetry required		Special case: Only half of the string is simulated, initial condition is symmetrical	
		Number of calculated points on the string (without the non-moving end points)	Number of integrators	Number of calculated points on the string (without the non-moving left end point)	Number of integrators
2	225.1 Hz (223.7 Hz meas.)	1	2 (1 THAT)	1	2 (1 THAT)
4	121.8 Hz (121.1 Hz meas.)	3	6 (2 THATs)	2	4 (1 THAT)
6	82.4 Hz (82.0 Hz meas.)	5	10 (2 THATs)	3	6 (2 THATs)
8	62.1 Hz (61.9 Hz meas.)	7	14 (3 THATs)	4	8 (2 THATs)
10	49.8 Hz (49.6 Hz meas.)	9	18 (4 THATs)	5	10 (2 THATs)
12	41.5 Hz (41.8 Hz meas.)	11	22 (5 THATs)	6	12 (3 THATs)
14	36.6 Hz	13	26 (6 THATs)	7	14 (3 THATs)
16	31.2 Hz (30.2 Hz meas.)	15	30 (6 THATs)	8	16 (4 THATs)
20	25.0 Hz	19	38 (8 THATs)	10	20 (4 THATs)
24	20.8 Hz	23	46 (10 THATs)	12	24 (5 THATs)
n	159.15 Hz · sqrt(2·(1-cos(180°/n)))	n-1	2n-2	n/2	n

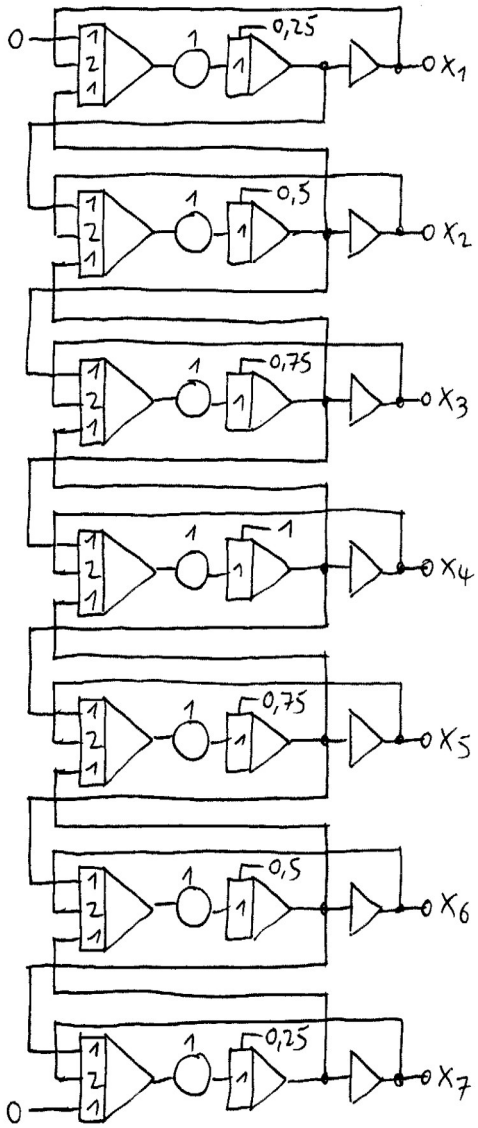
TC = 1 ms f = 1 / (TC · 2 · pi) = 159.15 Hz

Circuit 1: The guitar string is divided into 8 equal segments. The two end points are fixed, and the string elongation is calculated at 7 points.

Green: X_1 Magenta: X_2 Cyan: X_3 Yellow: X_4

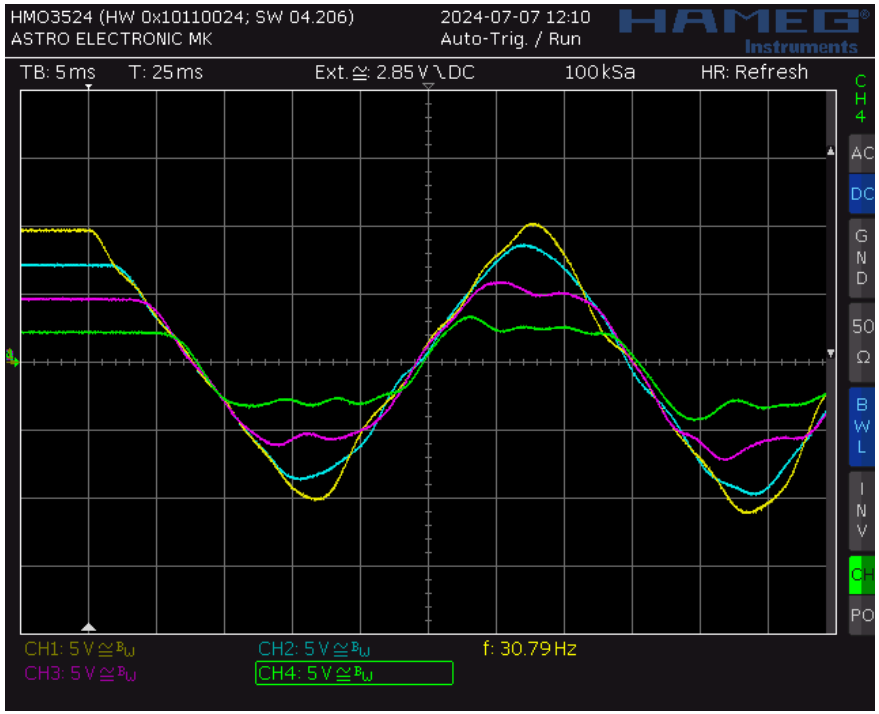


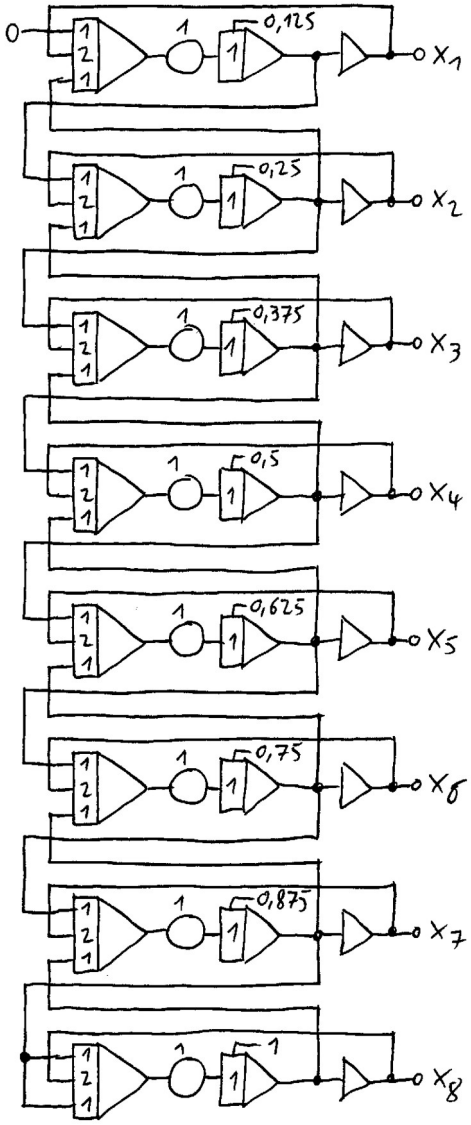




Circuit 2: If the initial elongation is made exactly in the middle of the string, the problem is symmetrical and only half of the string must be simulated. Half of the string is divided into 8 equal segments. The left point is fixed, and the string elongation is calculated at 8 points, the last one being in the center of the string. The result is much better that in the previous example.
The difference to the previous circuit is at the input of the last stage.

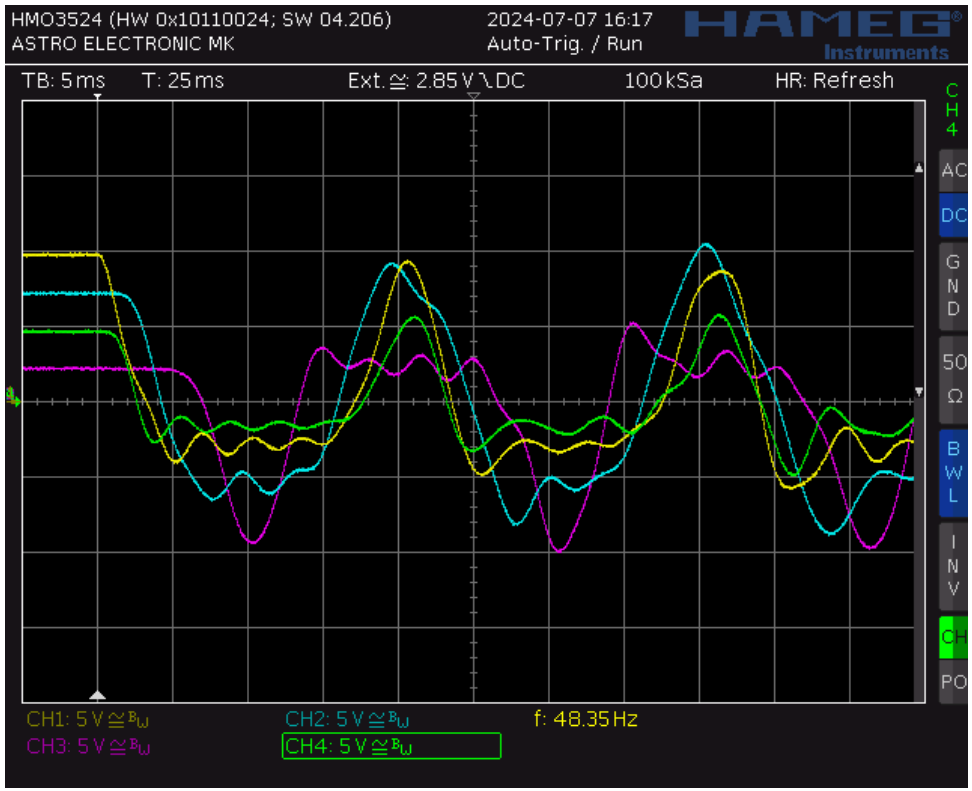
Green: X_2 Magenta: X_4 Cyan: X_6 Yellow: X_8

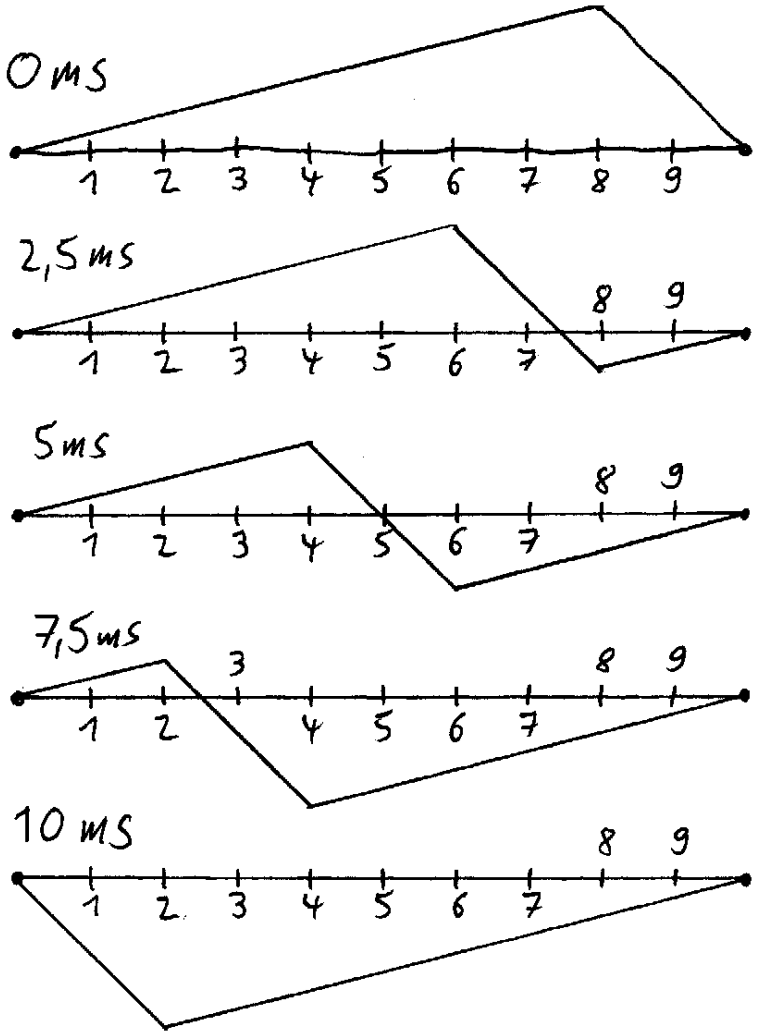




Circuit 3: This is an example where the initial elongation is not made in the middle of the string. Same circuit as circuit 1, but two more stages and different initial conditions.

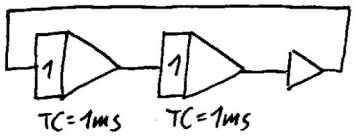
Magenta: X_2 Cyan: X_6 Yellow: X_8 Green: X_9



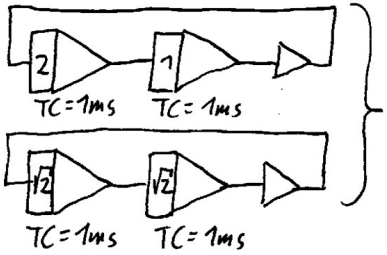


How can the frequency be calculated?

At the top are shown three simpler cases where we know the frequency.

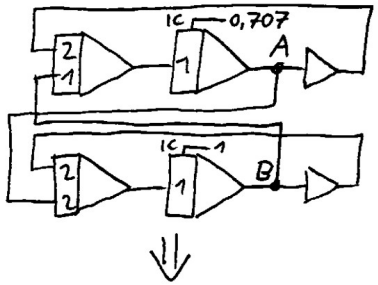


$$f = \frac{1}{1ms \cdot 2 \cdot \pi} = 159{,}15\ Hz$$



$$f = 159{,}15\ Hz \cdot \sqrt{2} = 225{,}07\ Hz$$

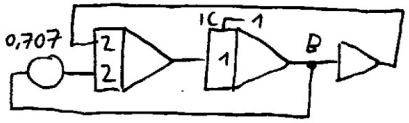
We can assume that signals A and B are in-phase sine waves. B is the position in the middle of the string, and A ist the position at 1/4 length.



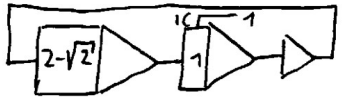
A and B are in-phase sine waves:

$$A = 0{,}707 \cdot B$$

With this assumption the upper part of the circuit can be eliminated and the frequency of the lower part can be calculated.



Confirmed by experiments !



$$f = 159{,}15\ Hz \cdot \sqrt{2-\sqrt{2}} = 121{,}81\ Hz$$

Where are the knots in the oscillation modes of the 12-segment string?

Waves	Freq [Hz]	Amplitude at the points on the 12-segment string												
		0	1	2	3	4	5	6	7	8	9	10	11	12
0.5	41.5	0	0.259	0.5	0.707	0.866	0.966	1	0.966	0.866	0.707	0.5	0.259	0
1	83	0	0.5	0.866	1	0.866	0.5	0	-0.5	-0.866	-1	-0.866	-0.5	0
1.5	124.5	0	0.707	1	0.707	0	-0.707	-1	-0.707	0	0.707	1	0.707	0
2	166	0	0.866	0.866	0	-0.866	-0.866	0	0.866	0.866	0	-0.866	-0.866	0
2.5	207.5	0	0.966	0.5	-0.707	-0.866	0.259	1	0.259	-0.866	-0.707	0.5	0.966	0
3	249	0	1	0	-1	0	1	0	-1	0	1	0	-1	0

Value = sin(Waves · 30° · n)

Where are the knots in the oscillation modes of the 16-segment string?

Waves	Freq [Hz]	Amplitude at the points on the 16-segment string															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0.5	30.2	0	0.195	0.383	0.556	0.707	0.831	0.924	0.981	1	0.981	0.924	0.831	0.707	0.556	0.383	0.195	0
1	60.4	0	0.383	0.707	0.924	1	0.924	0.707	0.383	0	-0.383	-0.707	-0.924	-1	-0.924	-0.707	-0.383	0
1.5	90.6	0	0.556	0.924	0.981	0.707	0.195	-0.383	-0.831	-1	-0.831	-0.383	0.195	0-707	0.981	0.924	0.556	0
2	120.8	0	0.707	1	0.707	0	-0.707	-1	-0.707	0	0.707	1	0.707	0	-0.707	-1	-0.707	0
2.5	151	0	0.831	0.924	0.195	-0.707	-0.981	-0.383	0.556	1	0.556	-0.383	-0.981	-0.707	0.195	0.924	0.831	0
3	181.2	0	0.924	0.707	-0.383	-1	-0.383	0.707	0.924	0	0.924	0.707	-0.383	-1	-0.383	0.707	0.924	0
3.5	211.4	0	0.981	0.383	-0.831	-0.707	0.556	0.924	-0.195	-1	-0.195	0.924	0.556	-0.707	-0.831	0.383	0.981	0
4	241.6	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0

Value = sin(Waves · 22.5° · n)

Where are the knots in the oscillation modes of the 20-segment string?

Waves	Freq [Hz]	Amplitude at the points on the 20-segment string																			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.5	25	0	0.16	0.31	0.45	0.59	0.71	0.81	0.89	0.95	0.99	1	0.99	0.95	0.89	0.81	0.71	0.59	0.45	0.31	0.16	0
1	50	0	0.31	0.59	0.81	0.95	1	0.95	0.81	0.59	0.31	0	-0.31	-0.59	-0.81	-0.95	-1	-0.95	-0.81	-0.59	-0.31	0
1.5	75	0	0.45	0.81	0.99	0.95	0.71	0.31	-0.16	-0.59	-0.89	-1	-0.89	-0.59	-0.16	0.31	0.71	0.95	0.99	0.81	0.45	0
2	100	0	0.59	0.95	0.95	0.59	0	-0.59	-0.95	-0.95	-0.59	0	0.59	0.95	0.95	0.59	0	-0.59	-0.95	-0.95	-0.59	0
2.5	125	0	0.71	1	0.71	0	-0.71	-1	-0.71	0	0.71	1	0.71	0	-0.71	-1	-0.71	0	0.71	1	0.71	0
3	150	0	0.81	0.95	0.31	-0.59	-1	-0.59	0.31	0.95	0.81	0	-0.81	-0.95	-0.31	0.59	1	0.59	-0.31	-0.95	-0.81	0
3.5	175	0	0.89	0.81	-0.16	-0.95	-0.71	0.31	0.99	0.59	-0.45	-1	-0.45	0.59	0.99	0.31	-0.71	-0.95	-0.16	0.81	0.89	0
4	200	0	0.95	0.59	-0.59	-0.95	0	0.95	0.59	-0.59	-0.95	0	0.95	0.59	-0.59	-0.95	0	0.95	0.59	-0.59	-0.95	0
4.5	225	0	0.99	0.31	-0.89	-0.59	0.71	0.81	-0.45	-0.95	0.16	1	0.16	-0.95	-0.45	0.81	0.71	-0.59	-0.89	0.31	0.99	0
5	250	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0

Value = sin(Waves · 18° · n)

Where are the knots in the oscillation modes of the 24-segment string?

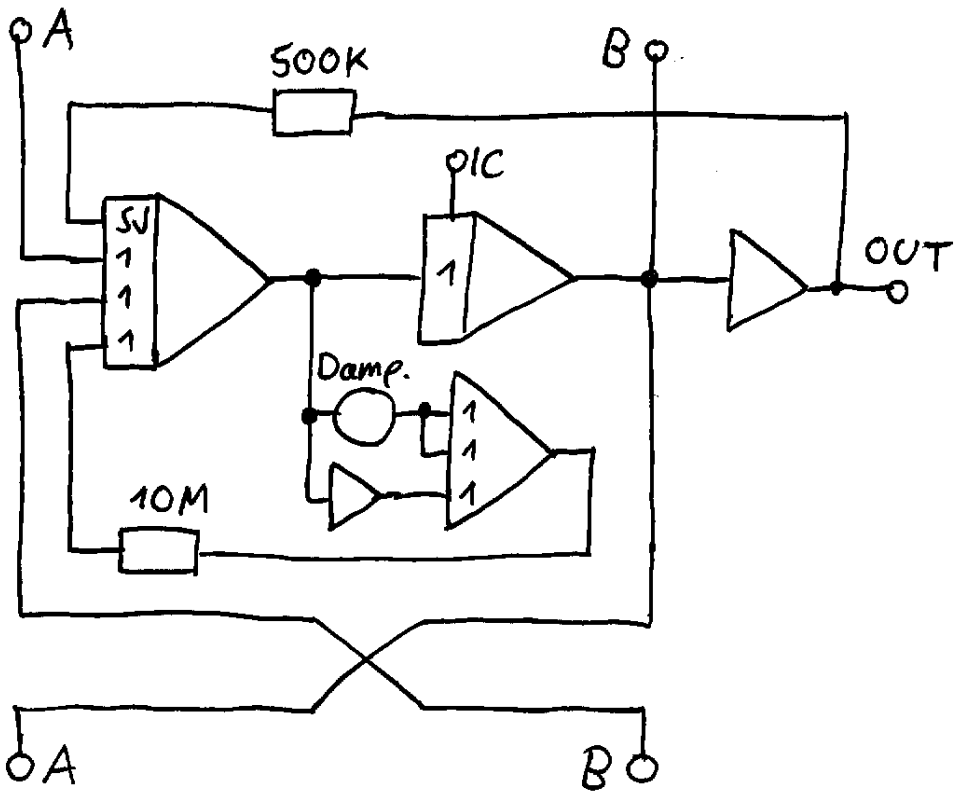
Waves	Freq [Hz]	Amplitude at the points on the 24-segment string																							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0.5	20.8	0	0.13	0.26	0.38	0.5	0.61	0.71	0.79	0.87	0.92	0.97	0.99	1	0.99	0.97	0.92	0.87	0.79	0.71	0.61	0.5	0.38	0.26	0.13	0
1	41.6	0	0.26	0.5	0.71	0.87	0.97	1	0.97	0.87	0.71	0.5	0.26	0	-0.26	-0.5	-0.71	-0.87	-0.97	-1	-0.97	-0.87	-0.71	-0.5	-0.26	0
1.5	62.4	0	0.38	0.71	0.92	1	0.92	0.71	0.38	0	-0.38	-0.71	-0.92	-1	-0.92	-0.71	-0.38	0	0.38	0.71	0.92	1	0.92	0.71	0.38	0
2	83.2	0	0.5	0.87	1	0.87	0.5	0	-0.5	-0.87	-1	-0.87	-0.5	0	0.5	0.87	1	0.87	0.5	0	-0.5	-0.87	-1	-0.87	-0.5	0
2.5	104	0	0.61	0.97	0.92	0.5	-0.13	-0.71	-0.99	-0.87	-0.38	0.26	0.79	1	0-79	0.26	-0.38	-0.87	-0.99	-0.71	-0.13	0.5	0.92	0.97	0.61	0
3	124.8	0	0.71	1	0.71	0	-0.71	-1	-0.71	0	0.71	1	0.71	0	-0.71	-1	-0.71	0	0.71	1	0.71	0	-0.71	-1	-0.71	0
3.5	145.6	0	0.79	0.97	0.38	-0.5	-0.99	-0.71	0.13	0.87	0.92	0.26	-0.61	-1	-0.61	0.26	0.92	0.87	0.13	-0.71	-0.99	-0.5	0.38	0.97	0.79	0
4	166.4	0	0.87	0.87	0	-0.87	-0.87	0	0.87	0.87	0	-0.87	-0.87	0	0.87	0.87	0	-0.87	-0.87	0	0.87	0.87	0	-0.87	-0.87	0
4.5	187.2	0	0.92	0.71	-0.38	-1	-0.38	0.71	0.92	0	-0.92	-0.71	0.38	1	0.38	-0.71	-0.92	0	0.92	0.71	-0.38	-1	-0.38	0.71	0.92	0
5	208	0	0.97	0.5	-0.71	-0.87	0.26	1	0.26	-0.87	-0.71	0.5	0.97	0	-0.97	-0.5	0.71	0.87	-0.26	-1	-0.26	0.87	0.71	-0.5	-0.97	0
5.5	228.8	0	0.99	0.26	-0.92	-0.5	0.79	0.71	-0.61	-0.87	0.38	0.97	-0.13	-1	-0.13	0.97	0.38	-0.87	-0.61	0.71	0.79	-0.5	-0.92	0.26	0.99	0
6	249.6	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0

Value = sin(Waves · 15° · n)

This is one stage of a guitar string simulator with adjustable damping.

Connect the signals A and B to the previous and next stage.

Coefficient	Damping
0	Strong damping
0.5	No damping
1	Negative damping (positive feedback)



An interesting thing can be simulated that's impossible with a real guitar string: In the simulation the string's end can be connected to its beginning. An impossible circular guitar string of infinite length!

The sound depends on:

- **The initial condition of the string points**
- **The damping coefficients**
- **All outputs of the stages have different signals.**

The clicking sound at the beginning comes from the 10ms when THAT is in IC mode. This can be avoided, if the initial condition is set to 0 for those stages that are used for the output.

In a real guitar, the initial condition is very simple: Both end points are fixed (0), and one point (not necessarily in the middle) is elongated (1). All other points are linearly interpolated.

However in this simulation, there are also other initial conditions possible. For example any points can be set to 0 (not only at the ends), and multiple points can be set to 1 or -1 simultaneously. It's impossible in a real guitar, because you can't release the string at multiple points exactly at the same time.

How to couple the output signals directly to the GH5S camera:

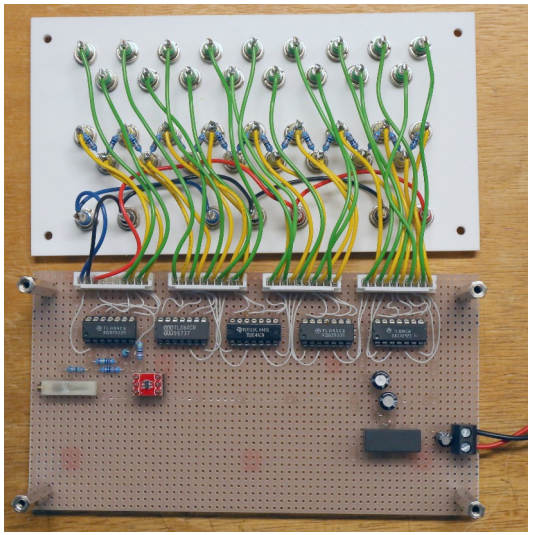
- **Connect the output of two stages to the X and Y outputs on THAT (+-10V Level).**
- **Connect the cinch outputs X and Y (+-1V Level) to the MIC input of the GH5S camera, using a normal 3.5mm stereo connector.**
- **In camera menue, set "Sound Rec Level Disp" to "ON", set "Sound Rec Level Adj." to 0dB, set "Sound Rec Level Limiter" to "ON" or "OFF", and set "Mic Socket" to "LINE". Save these settings.**

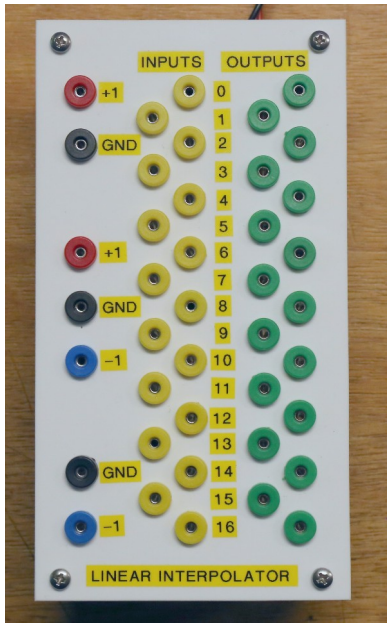
Open question: How can the fundamental frequency be calculated, if n stages are connected to a circular ring oscillator?

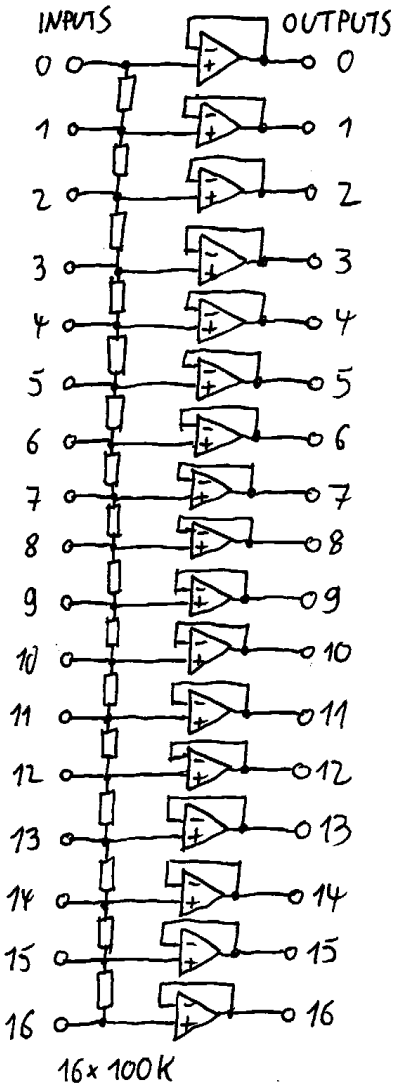
Number of stages n	Measured frequencies	Measured frequency · n ???
1		
2		
3		
4		
5	71.8 Hz	359 Hz
6		
7		
8		
9		
10	51.8 Hz	518 Hz
11		
12		
13	31.6 Hz	410.8 Hz
14	93.2 Hz = 3 · 31.1 Hz	434.9 Hz
15	27.9 Hz 87.7 Hz = 3 · 29.2 Hz	418.5 Hz 438.5 Hz

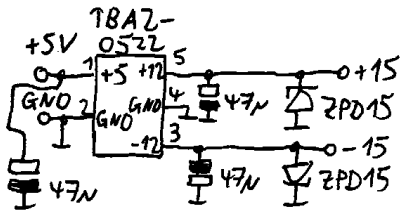
8.15.1 Linear Interpolator

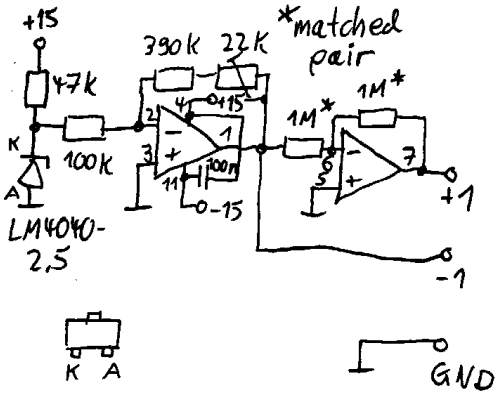
This is a circuit that simplifies making the initial conditions for the guitar string simulator.











5x TL084

8.16 Solid Bar Vibrations

https://wandinger.userweb.mwn.de/LA_Elastodynamik_2/kap_3_balken.pdf

https://wandinger.userweb.mwn.de/LA_Elastodynamik_2/index.html

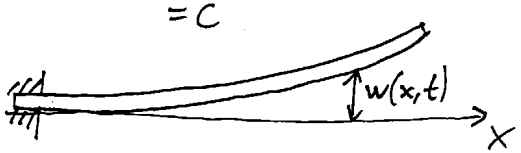
Bending of a beam, page 58: https://thesis.library.caltech.edu/5003/1/Glechorn%2Cjr_gj_1955.pdf

Giloi, Wolfgang / Lauber, Rudolf: Analogrechnen (german) page 270ff

Differential equation for the vibration of a bar: (from first link above)

$$\frac{d^4w}{dx^4} + \frac{\rho A}{E I_y} \cdot \frac{d^2w}{dt^2} = 0$$

$$\frac{d^4w}{dx^4} + \underbrace{\frac{\rho \cdot A}{E \cdot I_y}}_{=c} \cdot \frac{d^2w}{dt^2} = 0 \qquad (1)$$

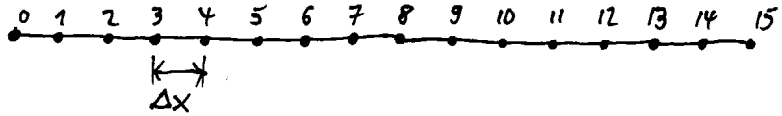


with w(x,t) = Deflection of the bar, perpendicular to the x axis

- x = Coordinate along the length of the bar
- t = time
- ρ = Density of the bar
- A = Cross section area
- E = Young's modulus (german: Elastizitätsmodul)
- I_y = Second moment of area (german: Flächenträgheitsmoment)

$$\frac{d^4w}{dx^4} \approx \frac{w_{n-2} - 4w_{n-1} + 6w_n - 4w_{n+1} + w_{n+2}}{\Delta x^4} \qquad (2)$$

$$\frac{w_{n-2} - 4w_{n-1} + 6w_n - 4w_{n+1} + w_{n+2}}{\Delta x^4} + c \cdot \frac{d^2w_n}{dt^2} = 0$$



$$\frac{d^2w}{dt^2} = \frac{-w_{n-2} + 4w_{n-1} - 6w_n + 4w_{n+1} - w_{n+2}}{c \cdot \Delta x^4} \qquad (3)$$

with $c = \rho A / (E I_y)$ (this is a constant)

and Δx is the horizontal distance from one point to the next

$$\frac{d^2 w_n}{dt^2} = \frac{-w_{n-2} + 4w_{n-1} - 6w_n + 4w_{n+1} - w_{n+2}}{c \cdot \Delta x^4}$$

I'm calculating the deflections at 15 equally spaced points w_1 to w_{15} which requires 30 integrators on 6 THATs. At the fixed (left) edge no calculation is required, because $w_0 = 0$.

How to handle the unknown deflection values, which don't exist at both ends of the beam?
See also Giloi, Wolfgang / Lauber, Rudolf: Analogrechnen (german) page 272ff

For the fixed (left) edge of the beam, I'm using $w_{-1} = 0$.

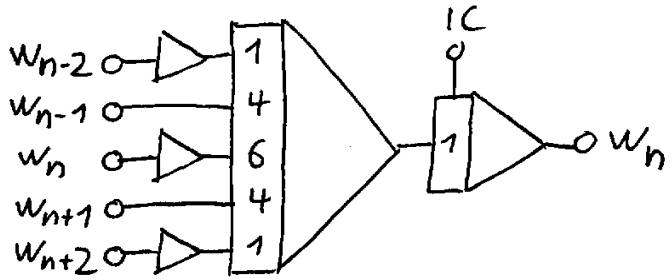
For the open (right) edge of the beam, I'm using a linear extrapolation of the slope between the known w_{14} and w_{15} deflections to get the unknown deflections w_{16} and w_{17}:

$$w_{16} = w_{15} + (w_{15} - w_{14}) = 2\,w_{15} - w_{14} \qquad w_{17} = w_{15} + 2\,(w_{15} - w_{14}) = 3\,w_{15} - 2\,w_{14}$$

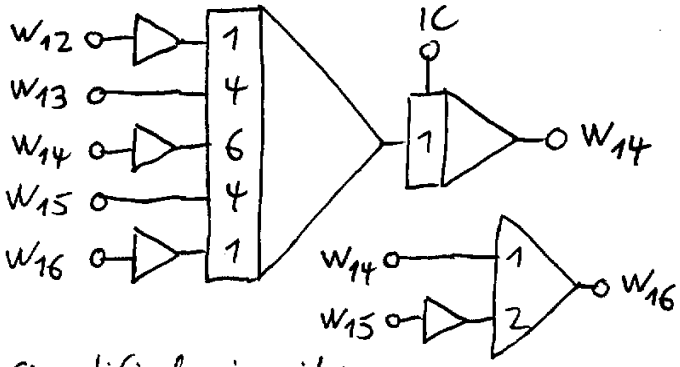
Initial conditions from C# program:

	Oscillation order			
Points	**1**	**2**	**3**	**4**
w1	-0.008	-0.044	-0.113	-0.203
w2	-0.029	-0.154	-0.359	-0.560
w3	-0.064	-0.301	-0.605	-0.754
w4	-0.110	-0.455	-0.746	-0.620
w5	-0.166	-0.590	-0.722	-0.195
w6	-0.230	-0.683	-0.526	0.316
w7	-0.301	-0.720	-0.203	0.659
w8	-0.379	-0.690	0.163	0.662
w9	-0.461	-0.589	0.474	0.326
w10	-0.547	-0.423	0.644	-0.171
w11	-0.635	-0.199	0.620	-0.567
w12	-0.725	0.070	0.395	-0.643
w13	-0.817	0.368	0.007	-0.330
w14	-0.908	0.682	-0.480	0.277
w15	-1.000	1.000	-1.000	1.000

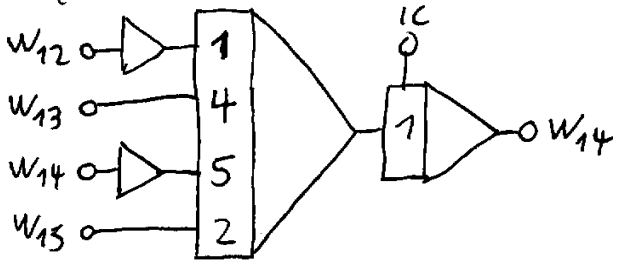
For points w_1 to w_{13}:



For point w_{14}:

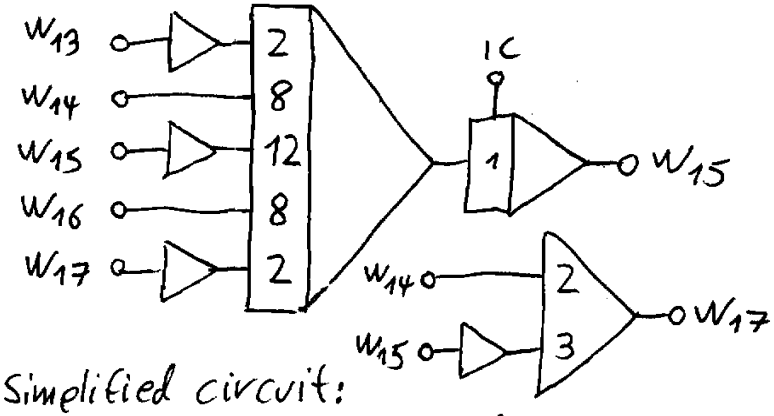


Simplified circuit:

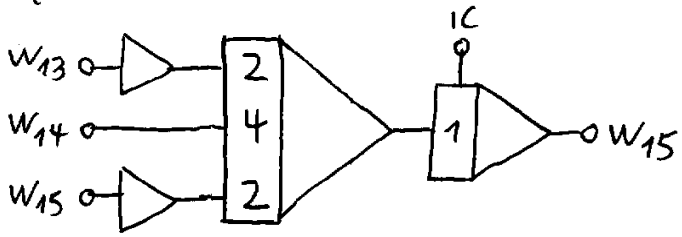


Double weights, because mass is only half at w_{15}

For point w_{15}: ↓



Simplified circuit:



C# Program for calculating the initial conditions of the vibrating beam. The number of points and the oscillation order can be changed.

```
using System;
using System.Windows.Forms;
using System.Globalization;

namespace Balkenschwingung
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            calc();
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {  calc();  }

        private void numericUpDown2_ValueChanged(object sender, EventArgs e)
        {  calc();  }

        void calc()
        {
            CultureInfo invC = CultureInfo.InvariantCulture;
            double k, x, w, gamma;
            int nu = (int)numericUpDown1.Value;   // order 1, 2, 3 ...
            int n = (int)numericUpDown2.Value;    // number of points, excluding the point at the left edge

            switch (nu)
            {
                case 1:  k = 1.8751; break;
                case 2:  k = 4.6941; break;
                case 3:  k = 7.8548; break;
                default: k = (2.0 * (double)nu - 1.0) * Math.PI / 2.0; break;
            }
            gamma = (Math.Cos(k) + Math.Cosh(k)) / (Math.Sin(k) + Math.Sinh(k));
            richTextBox1.Clear();
            for (int i = 1; i <= n; i++)
            {
                x = (double)i / n;    // 0 to 1
                w = 0.5 * (Math.Cos(k * x) - Math.Cosh(k * x) - gamma * (Math.Sin(k * x) - Math.Sinh(k * x)));
                richTextBox1.AppendText(w.ToString("F3", invC) + "\n");
            }
        }
    }
}
```

8.17 Plate Vibrations

https://pkel015.connect.amazon.auckland.ac.nz/SolidMechanicsBooks/Part_II/06_PlateTheory/06_PlateTheory_08_Vibrations.pdf

Bending of a plate, page 62: https://thesis.library.caltech.edu/5003/1/Glechorn%2Cjr_gj_1955.pdf

8.18 Euler Spiral

See chapter 9.5 in THAT booklet: https://the-analog-thing.org/THAT_First_Steps.pdf

See also: https://the-analog-thing.org/docs/dirhtml/rst/applications/euler_spiral/alpaca_33/
This is possibly not a real Euler spiral.

Notes from Matthias Weiss, Source: https://www.facebook.com/groups/theanalogthing/?multi_permalinks=837385031817621

"My least favorite section in the First Steps booklet is the Euler spiral. Instead of teaching the reader the strategy behind setting up the problem, and which is transferable to similar problems, it simply offers a recipe one can imitate. Here is the process:

In order to generate the two integrants, *find out how to rephrase the problem as a system of differential equations*. Take derivatives and, by inspection, discover that they satisfy

x'(t) = –t y(t)

y'(t) = t x(t).

Turn the independent variable "t" into a voltage **by integrating a constant with suitable IC, thus creating a ramp from –1 to 1 which reflects the passage of time in "voltage space."**

Calculate x(t) and y(t) by integrating, then integrate once more to get the parametric representation of the curve itself.

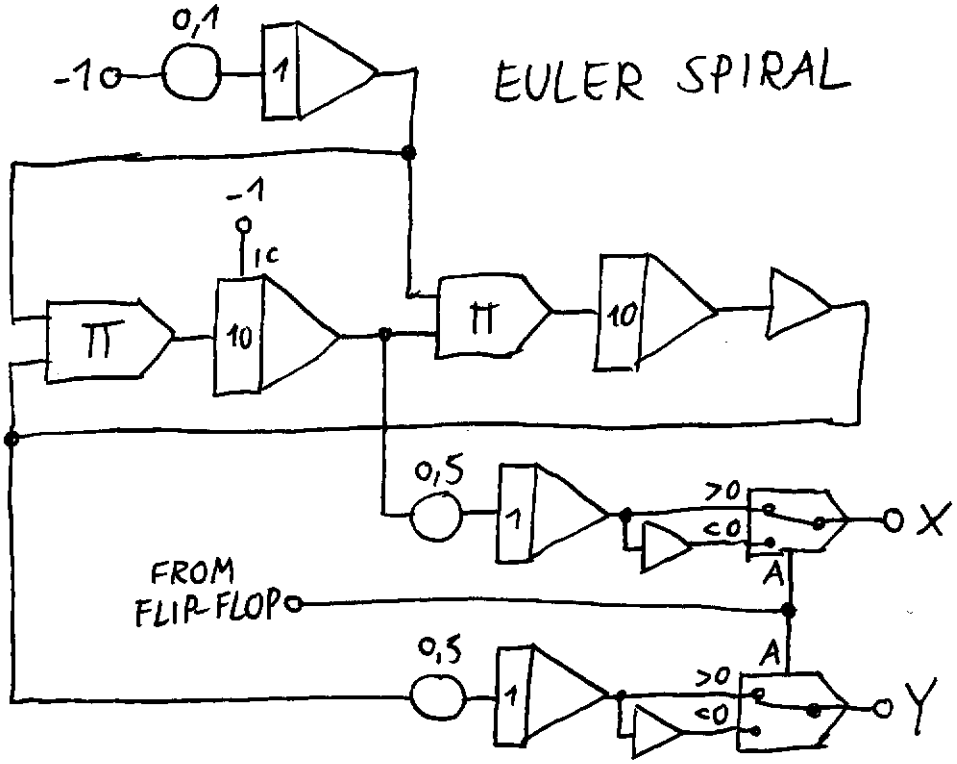
Include scaling and appropriate IC where needed.

Once I had the system of differential equations in front of me, I could just patch it up by looking at the math...and somehow I got a nice picture using REP rather than REPF as recommended in the booklet. "

See also: https://www.facebook.com/groups/305923044963825/?multi_permalinks=1002327691990020

See also this PDF from Matthias Weiss: https://www.facebook.com/download/4065910663692790/Two-Branch-Euler-Spiral.pdf

If a Flip-Flop is connected to the hybrid port, the Flip-Flop's output can be used to toggle between the two branches of the Euler Spiral.



8.19 Chaotic Systems

See also: Wikipedia "List of Chaotic Maps" https://en.wikipedia.org/wiki/List_of_chaotic_maps

See also: Wikipedia category "Chaotic Maps" https://en.wikipedia.org/wiki/Category:Chaotic_maps

See also: https://www.dynamicmath.xyz/strange-attractors/
(contains Lorenz attractor, Thomas attractor, Langford (Aizawa) attractor, Dadras attractor, Chen attractor, Lorenz83 attractor, Rössler attractor, Halvorsen attractor, Rabinovich-Fabrikant attractor, Three-Scroll Unified Chaotic System, Sprott attractor, Four-Wing attractor)

See also: https://www.bentamari.com/attractors.html
(contains Double pendulum, Duffing attractor, Henon attractor, Swinging Atwood's machine, Tinkerbell attractor, KAM Torus (Kolmogorov, Arnold, Moser), de Jong attractor, Chua attractor, Ikeda attractor, Lorenz attractor, Pickover attractor, Rössler attractor, Tamari attractor)

Sorted alphabetically:

Aizawa attractor https://analogparadigm.com/downloads/alpaca_17.pdf
see also: https://www.dynamicmath.xyz/strange-attractors/

Brusselator https://analogparadigm.com/downloads/alpaca_40.pdf
Wikipedia: https://en.wikipedia.org/wiki/Brusselator

Celestial mechanics see Three-body problem

Chen Attractor https://www.facebook.com/groups/305923044963825/?multi_permalinks=990229523199837
see also: https://www.dynamicmath.xyz/strange-attractors/
see also: https://analogparadigm.com/downloads/alpaca_52.pdf

Chen-Celikovsky system http://lsc.amss.ac.cn/~ljh/02LC.pdf

Chen-Lü system (same as above?)

Chua oscillator https://analogparadigm.com/downloads/alpaca_3.pdf
Wikipedia "Chua's Circuit" https://en.wikipedia.org/wiki/Chua's_circuit
see also: http://www.scholarpedia.org/article/Chua_circuit

Competitive Lotka-Volterra equations https://en.wikipedia.org/wiki/Competitive_Lotka%E2%80%93Volterra_equations
See also "Generalized Lotka-Volterra equation"

Dadras attractor, version of Matthias Weiss: https://www.facebook.com/groups/305923044963825?multi_permalinks=929084782647645
see also: https://www.dynamicmath.xyz/strange-attractors/
see also: https://analogparadigm.com/downloads/alpaca_46.pdf

Double pendulum https://the-analog-thing.org/docs/dirhtml/rst/applications/double_pendulum/alpaca_21/
se also: https://en.wikipedia.org/wiki/Double_pendulum

Duffing oscillator https://analogparadigm.com/downloads/alpaca_29.pdf
Wikipedia: https://en.wikipedia.org/wiki/Duffing_equation

Elastic Pendulum https://en.wikipedia.org/wiki/Elastic_pendulum

Elegant chaos https://analogparadigm.com/downloads/alpaca_15.pdf
See also: https://the-analog-thing.org/docs/dirhtml/rst/applications/elegant_chaos/alpaca_15/

Four-wing attractor https://analogparadigm.com/downloads/alpaca_45.pdf
see also: https://www.dynamicmath.xyz/strange-attractors/

Generalized Lorenz-like system http://lsc.amss.ac.cn/~ljh/04LCC.pdf

Generalized Lotka-Volterra equation https://en.wikipedia.org/wiki/Generalized_Lotka–Volterra_equation
See also "Competitive Lotka-Volterra equations"

Genesio-Tesi chaotic attractor http://lsc.amss.ac.cn/~ljh/04LCC.pdf

Hadamard's dynamical system https://en.wikipedia.org/wiki/Hadamard's_dynamical_system

Hadley chaotic circulation https://en.wikipedia.org/wiki/Hadley_cell

Halvorsen attractor https://www.dynamicmath.xyz/strange-attractors/
see also: https://analogparadigm.com/downloads/alpaca_46.pdf

Henon-Heiles system: https://en.wikipedia.org/wiki/Hénon–Heiles_system
Reduced Henon-Heiles system https://analogparadigm.com/downloads/alpaca_46.pdf

Hindmarsh-Rose-model of neuronal bursting https://analogparadigm.com/downloads/alpaca_28.pdf
see also here: https://the-analog-thing.org/docs/dirhtml/rst/applications/hindmash_rose_neuron/spiking_neuron/
Wikipedia: https://en.wikipedia.org/wiki/Hindmarsh–Rose_model
Hindmarsh-Rose with feedback https://analogparadigm.com/downloads/alpaca_41.pdf

Hyperchaotic Chen system https://web.archive.org/web/20120530145100/http://www.ijest.info/docs/IJEST11-03-05-262.pdf
see also: https://www.sciencedirect.com/science/article/abs/pii/S0096300304006988
see also: https://www.researchgate.net/publication/243333058_Analysis_of_the_hyper-chaos_generated_from_Chen's_system

Hyperion (a Saturn moon with chaotic rotation) https://en.wikipedia.org/wiki/Hyperion_(moon)

Hyper-Rössler chaotic attractor http://www.scholarpedia.org/article/Hyperchaos

Ikeda chaotic attractor https://www.bentamari.com/attractors.html

Knot-Holder chaotic oscillator http://www.scholarpedia.org/article/Chaos_topology

Kuramato-Sivashinsky equation https://en.wikipedia.org/wiki/Kuramoto–Sivashinsky_equation

Langford attractor https://www.dynamicmath.xyz/strange-attractors/

Li system https://analogparadigm.com/downloads/alpaca_54.pdf
Note from Matthias Weiss: "While the third multiplier did not pose a problem, having to use nine pots did, so I did another time scaling (which is the proper context for "global scaling"). That killed the two 0.4 coefficients and made the tiny 0.01 coefficient a bit larger.
Why is it a "time scaling"? Because unlike Engineering World, in Math World we can't just multiply one side of the equation by λ; but what we can do is introduce a new time variable τ satisfying dt/dτ = λ, so that on all the left-hand sides multiplying gives d/dt dt/dτ = d/dτ , but on the right-hand sides we multiply by λ."
Note from me: Factors -0.1, -0.2 and -0.4 can be realized without coefficients, using open amplifiers.

Lorenz attractor https://analogparadigm.com/downloads/alpaca_2.pdf
see also here: https://the-analog-thing.org/docs/dirhtml/rst/applications/lorenz/alpaca_2/
see also: https://www.dynamicmath.xyz/strange-attractors/
Wikipedia: https://en.wikipedia.org/wiki/Lorenz_system
see also here, where the 3 variables are described as "stream function", "change in temperature" and "deviation in linear temperature":
https://www.facebook.com/photo/?fbid=1212783870264878&set=a.167457014797574
Yet another Lorenz system https://analogparadigm.com/downloads/alpaca_16.pdf
see also: "Generalized Lorenz-like system"

Lorenz83 attractor https://www.dynamicmath.xyz/strange-attractors/

Mackey-Glass equations https://en.wikipedia.org/wiki/Mackey–Glass_equations

Mathieu's equation https://analogparadigm.com/downloads/alpaca_10.pdf
and also here: https://the-analog-thing.org/docs/dirhtml/rst/applications/mathieu/alpaca_10/
Mathieu's equation revisited (with hybrid computer) https://analogparadigm.com/downloads/alpaca_47.pdf
Wikipedia: https://en.wikipedia.org/wiki/Mathieu_function

Mixmaster Universe https://en.wikipedia.org/wiki/Mixmaster_universe

Moore-Spiegel attractor https://demonstrations.wolfram.com/MooreSpiegelAttractor/

Nonlinear chaos https://analogparadigm.com/downloads/alpaca_8.pdf
see also here: https://the-analog-thing.org/docs/dirhtml/rst/applications/nonlinear_chaos/alpaca_8/

Oregonator https://analogparadigm.com/downloads/alpaca_51.pdf
Wikipedia: https://en.wikipedia.org/wiki/Oregonator

Rabinovich-Fabrikant equations https://en.wikipedia.org/wiki/Rabinovich–Fabrikant_equations
see also: https://www.dynamicmath.xyz/strange-attractors/

Raleigh-Bénard convection https://en.wikipedia.org/wiki/Rayleigh–Bénard_convection

Rikitake attractor https://web.archive.org/web/20100620195842/http://www.ams.jhu.edu/~castello/391/articles/rikitake.pdf

Rössler attractor https://analogparadigm.com/downloads/alpaca_1.pdf
see also here: https://the-analog-thing.org/docs/dirhtml/rst/applications/roessler_attractor/aplaca_1/
see also: https://www.dynamicmath.xyz/strange-attractors/
Wikipedia: https://en.wikipedia.org/wiki/Rössler_attractor

Rucklidge system http://www.wseas.us/e-library/conferences/2011/lasi/DYMANOW/DYMANOW-17.pdf

Shaw attractor https://math.la.asu.edu/~gardner/Lorenz+Shaw.pdf

Sprott Chaotic Jerk Circuit https://sprott.physics.wisc.edu/pubs/paper352.pdf

Sprott SQ_F system https://analogparadigm.com/downloads/alpaca_46.pdf

Sprott SQ_M model https://analogparadigm.com/downloads/alpaca_31.pdf
see also here: https://the-analog-thing.org/docs/dirhtml/rst/applications/sqm_model/alpaca_31/
Chaotic Sprott system https://analogparadigm.com/downloads/alpaca_43.pdf
see also: https://www.facebook.com/groups/305923044963825?multi_permalinks=930263315863125
see also: https://www.dynamicmath.xyz/strange-attractors/

Sprott Symmetric Chaotic Flow https://sprott.physics.wisc.edu/chaos/symmetry.htm

Strizhak-Kawczynski chaotic oscillator https://web.archive.org/web/20151222080355/http://jlswbs.blogspot.de/2011/10/strizhak-kawczynski.html

Thomas' cyclically symmetric attractor https://en.wikipedia.org/wiki/Thomas'_cyclically_symmetric_attractor
see also: https://www.dynamicmath.xyz/strange-attractors/
see also: https://analogparadigm.com/downloads/alpaca_53.pdf

Three-body problem https://analogparadigm.com/downloads/alpaca_11.pdf
Wikipedia: https://en.wikipedia.org/wiki/Three-body_problem

Three-Scroll Unified Chaotic System https://www.dynamicmath.xyz/strange-attractors/

Van der Pol oscillator https://en.wikipedia.org/wiki/Van_der_Pol_oscillator

8.20 Nuclear Decay Chain

Bateman equation: https://en.wikipedia.org/wiki/Bateman_equation

See also: https://www.facebook.com/groups/theanalogthing/posts/905830248306432

8.21 Legendre Polynomials

https://en.wikipedia.org/wiki/Legendre_polynomials

See also: https://www.facebook.com/groups/theanalogthing/posts/939076381648485/
and https://www.facebook.com/groups/theanalogthing/posts/939676278255162/

This is copied from the PDF from Matthias Weiss (this is complicated mathematical stuff, I didn't understand it but just want to save it for later):

Analog Computing
Time Scaling and Legendre Equation

Legendre's equation reads

$$\left(1-t^2\right)\ddot{x}-2t\dot{x}+n(n+1)x=0,$$

where customarily $n\in\mathbb{N}$, but that is not a requirement. To transform into a system of first-order equations, let $\dot{x}=y$, so we get

$$\begin{cases}\dot{x}=y\\ \left(1-t^2\right)\dot{y}-2ty+n(n+1)x=0.\end{cases}$$

Solving for the first derivatives, this becomes

$$\begin{cases}\dot{x}=y\\ \dot{y}=\dfrac{2ty-n(n+1)x}{1-t^2}\end{cases}$$

We introduce the new time variable s via a – yet to be determined – relationship $t=t(s)$. Then we can multiply both sides of each equation by $t'=\dfrac{dt}{ds}$ and obtain

$$\begin{cases}\dfrac{dt}{ds}\dfrac{dx}{dt}=\dfrac{dt}{ds}y\\ \dfrac{dt}{ds}\dfrac{dy}{dt}=\dfrac{dt}{ds}\dfrac{2ty-n(n+1)x}{1-t^2}\end{cases}$$

Recognize the chain rule on the left-hand side and notice that setting $\dfrac{dt}{ds}=1-t^2$ results in

$$\begin{cases}\dfrac{dx}{ds}=\left(1-t^2\right)y\\ \dfrac{dy}{ds}=2ty-n(n+1)x\\ \dfrac{dt}{ds}=1-t^2\end{cases}$$

with initial conditions $t(0)=-1$. Plot $(t(s),x(s))$ in XY-mode. No more quotient!!

To set it up, implement the equation for $t(s)$ and test it in $REPF$ mode. I set the initial condition to $t(0)=-1$, but in order to avoid the singularity, I used a pot to get as close to -1 as possible without making the circuit angry. You should get a nice smooth ramp from $t=-1$ to $t=1$ if you set the IC and the execution time appropriately.

You'll need two more multipliers for $(1-t^2)y$ and for ty, and in order to get a nice range for $n(n+1)$, use a $10\,\mathrm{k}\Omega$ resistor into the summing junction of y.

I set up $x(0)$ to be variable between -1 and 1, though the circuit does best with a modest initial condition. I set $y(0)=0$; i.e. left it open.

8.22 Bessel Function

https://en.wikipedia.org/wiki/Bessel_function

See also: https://analogparadigm.com/downloads/alpaca_35.pdf
See also: https://www.facebook.com/groups/theanalogthing/posts/939783401577783/

The following text is from the Facebook posting of Matthias Weiss:

The differential equation is

$t^2 x'' + t x' + (t^2 – n^2) x = 0$

That's what is in the PDF (for Legendre polynomials, see above):
Write as system of first-order ode and multiply both sides of both equations (x'(t)=..., y'(t)=...) by dt/ds (t: old time, s: new time). On the left hand side, you will encounter chain rules (dx/dt*dt/ds=dx/ds); on the right hand side, choose dt/ds to make the denominators go away. In this case, dt/ds = t^2. With respect to the new time s, this system is autonomous, but "t" is still the time of the system. That is why you plot (t(s), x(s)) parametrically.

Introducing the new time variable that spares the division-by-electronics (and renaming "s" back to "t") gives

$x' = t^2 y$
$y' = (n^2 - t^2) x – t y$
$t' = t^2$

This yields nothing discernible because the "periods" of the Bessel functions are too large to accommodate a t-interval of [0,1], so we compress t by a factor of 10 using ordinary scaling:

$x' = t^2 y$
$y' = (0.01n^2 - t^2) x – 0.1 t y$
$t' = t^2$

Since I divided all coefficients by 100 so as to be able to use my pots, that factor got stuffed into the time factor, so it's a bit of a sluggish circuit.

Graph (t, x) parametrically (in XY-mode). The functions are no visual spectacle by a long shot (even for various values of n), but it's nice to see graphs roughly as would be expected(*). The scope image is DC coupled, with Y = 0 set ot the horizontal axis, and t=0 has been moved to the left. This is 50mV per box in each direction.

...

I think your biggest problem is that you try to read dense math by staring at it, rather than following step by step with paper and pencil. While the idea comes from an engineering text, I too followed along with paper and pencil to justify each step in my mind. I have to assume that the individual steps

are well within the scope of what an engineer learns in their math courses, so let me get started with what I think is the absolute minimum:

(1) I have to believe that you are able to turn the second-order equation into a system of first order equations:
$x' = y$
$y' = f(t, x, y)$

(2) I must also believe that you know that you can multiply both sides of any equation by the same number:
$c*x' = c*y$
$c*y' = c*f(t, x, y)$

(3) If you did (1) out on paper, your f(t, x, y) will have a denominator. What should "c" be so that c*f(t, x, y) doesn't have a denominator anymore?

(4) What should "c" on the left be so that the derivative is with respect to a new time variable s? Keep in mind the chain rule.

Let's try it:

The given equation is $t^2 x'' + t x' + (t^2 - n^2) x = 0$

Step 1:

$x' = dx/dt = y$
$t^2 y' + t y + (t^2 - n^2) x = 0$ --> $y' = dy/dt = (-t y - (t^2 - n^2) x) / t^2$

Step 2 and 3:

$c = t^2$
$c x' = c\ dx/dt = c y$
$c y' = c\ dy/dt = -t y - (t^2 - n^2) x$

At the end of Step 3, I get
$c*[dy/dt] = c*[(-t y - (t^2 - n^2) x) / t^2]$
and so if $c = t^2$,
$c*dy/dt = -t y - (t^2 - n^2) x;$
that's it. You don't need to substitute the t because it will be determined by the next condition: switching the derivative with respect to t to the derivative with respect to s on the left-hand side. This is where the chain rule comes in:
$dy/ds = dy/dt * dt/ds = dt/ds * dy/dt.$
We now know that, in order to make this work, $c = dt/ds$. We also know from cleaning up the denominator on the right hand side, that $c = t^2$. Thus we get the differential equation $dt/ds = t^2$, which we add to the two we already have.

If you solve a differential equation, you don't ask for the value of the independent variable t, you ask for the functions x(t) and y(t), so "s = ???" is an unnecessary question because s is the new independent variable. For example, if you solve
x'=x,
you get
x(t) = c*e^t,
and you don't ask "t = ???"! So there is no need to ask for s if it becomes the new "time." However, even though "s" is the new time variable, we are still interested in x(t), because the s-time is distorted. That is why we solve for t(s) and x(s) and plot both in XY-mode, rather than displaying x(s) in "scroll" mode, say.

Step 4:

$c = t^2$
$c\ dx/dt = c\ y$
$c\ dy/dt = -t\ y - (t^2 - n^2)\ x$

$c = dt/ds$

$dt/ds = t^2$

$dt/ds\ dx/dt = c\ y$
$dt/ds\ dy/dt = -t\ y - (t^2 - n^2)\ x$

$dt/ds = t^2$
$dx/ds = t^2\ y$
$dy/ds = -t\ y + (n^2 - t^2)\ x$

Well, I can follow the procedure step by step now, but I can't say that I really understood it.

Open questions:

How must the initial conditions for x, y and t be set?

How do we know in which direction t will run over time s?

Let's assume we want to plot the frunction from t1 to t2. If we start at t1, isn't it possible that t begins to run in the correct direction towards t2, but then changes direction so that t2 is never reached?

Another thought: A more familiar process of nonlinear time scaling is "integration by substitution". For example, to solve (which is a simple case of a differential equation)
$x'(t) = t \sin(t^2)$,
you set $u(t) = t^2$, $du/dt = 2t$. Our process here is quite similar, only you get your change of variables *implicitly* via a differential equation (the one we add to the set). For this problem, you simply calculate
$\int \frac{1}{2} \sin(u)\, du$,
but since you are polite, you write the antiderivative back in terms of t for the reader. In our case, we calculate the solution (via THAT) in terms of s, but present it on the scope in terms of t.

$x'(t) = dx/dt = t \sin(t^2)$

$u(t) = t^2$ $du/dt = 2\,t$

$dt/du * dx/dt = dt/du * t \sin(t^2)$

$dx/du = 1/2 * \sin(t^2)$???

$dt/du = 1/(2\,t)$???

I don't yet see how this is better than the original equation. It's all totally unclear.

Shouldn't this be the solution? $x(t) = -1/2 \cos(t^2)$

Related: Chapter 11.6 "Lösung von Differentialgleichungen" on page 783 in

Tietze, U. and Schenk, Ch.: Halbleiter-Schaltungstechnik, 11. Auflage, ISBN 3-540-64192-0

8.23 Diffusion in Meteoroloy

Brock, Fred V.: Analog Computing Techniques Applied to Atmospheric Diffusion: Continuous Line Source
https://journals.ametsoc.org/view/journals/apme/1/4/1520-0450_1962_001_0444_actata_2_0_co_2.xml

8.24 Fractals with analog Computers

https://www.youtube.com/watch?v=Pv26QAOcb6Q

9 Lucidac

Lucidac: https://anabrid.com/lucidac

Lucidac Handbook: https://anabrid.dev/docs/lucidac-user-docs.pdf

Lucidac firmware (Teensy 4.1): https://github.com/anabrid/lucidac-firmware

Is this the documentation for the software running on Teensy 4.1? I'm not sure: https://anabrid.dev/docs/hybrid-controller/index.html

This file contains some hints which chips are used: https://github.com/anabrid/lucidac-firmware/tree/main/lib/hal-lucidac/src/chips

Lucipy documentation: https://anabrid.dev/docs/lucipy/dirhtml/

Lucipy examples: https://github.com/anabrid/lucipy/tree/master/examples/rev1-final

Github: https://github.com/anabrid/lucipy

How to install Lucipy without using git:

- **Go to https://github.com/anabrid/lucipy and then click on "<> Code" --> Download ZIP.**
- **Extract the content of the ZIP file to a folder.**

Alternatively, after Python has already been installed, Lucipy can be installed by typing this in a command line:

pip install lucipy

Lucigui, a GUI for the analog-digital hybrid computer LUCIDAC:

Github: https://github.com/anabrid/lucigui

(extremely complicated to install, I didn't even try)

Online version of Lucigui (this does work, but it was developed with an older version of the Lucidac hardware, so that the JSON output file isn't compatible with the newer hardware):

https://lucidac.online/main/ (Click on "Editor")

Components can be deleted with backspace.

Double-click on a coefficient to enter a value.

The exported JSON file seems to consist of two parts: "SvelteFlowView" describes the graphical schematic diagram (this is irrelevant for configuring the analog computer) and "ClusterConfig" contains the informations for configuring the analog computer.

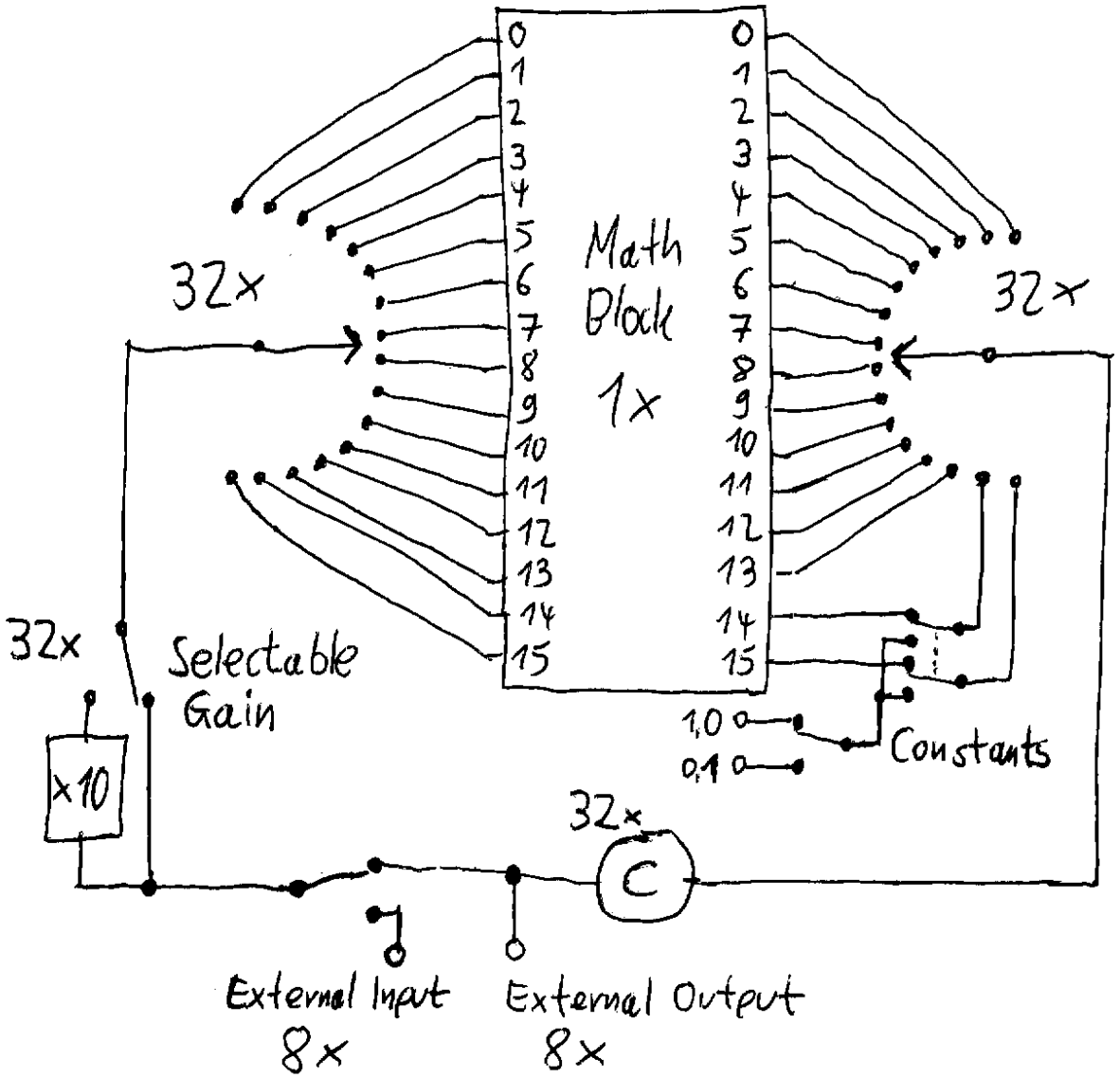
Install Python: https://www.python.org/ (Windows Installer 64-bit)

Run the windows installer "python-3.12.6-amd64.exe" and then it's important to tick the box "Add python.exe to path" before running the default installation.

Simplified block diagram of Lucidac. It's simply a 16-channel math block surrounded by 32 freely configurable feedback paths. All inputs of the math block have implicit summing of up to 32 feedback paths. Outputs 14 and 15 of the math block can be replaced by a constant source (simplified in this drawing).

Some answers to my questions:

- **Implicit summing does not reverse the sign**
- **Available constants are 0.1 and 1, but they can't be used simultaneously in a circuit. Either 0.1 or 1.0.**
- **Coefficients can be set in the range of [-10;10] while it activates/deactivates a factor switch, the purpose is extending the coefficient resolution over one decade more, this increases the dynamic range and corresponding resolution, usually you loose 3.2 bits per decade - no resolution loss for the first decade**



9.1 How to create a JSON File from a *.py Example File

Step 1: Open a command line window.

Step 2: Navigate to the folder where the *.py example is.

Step 3: Open the *.py file in an editor, for example lorenz.py :

```
# Implement Lorentz attractor as in
# https://analogparadigm.com/downloads/alpaca_2.pdf

from lucipy import LUCIDAC, Circuit
from time import sleep

a = 1.0
b = 2.8
c = 2.666 / 10

l = Circuit()                           # Create a circuit

mx = l.int(ic = -0.1)
my = l.int(ic = 0.3)
mz = l.int(ic = 0.5)
xz = l.mul()
xy = l.mul()

l.connect(mx, xz.a)                     # Product -x * -z = xz
l.connect(mz, xz.b)

l.connect(mx, xy.a)                     # Product -x * -y = xy
l.connect(my, xy.b)

l.connect(my, mx, weight = -a)
l.connect(mx, mx, weight = a*0.8)       # 0.8 increases stability

l.connect(mx, my, weight = -b)
l.connect(xz, my, weight = -5)
l.connect(my, my, weight = 0.1)

l.connect(xy, mz, weight = 5)
l.connect(mz, mz, weight = c)
```

```
l.probe(mx, front_port=0)
l.probe(my, front_port=1)
l.probe(mz, front_port=2)

hc = LUCIDAC()

hc.set_circuit(l)

hc.manual_mode("ic")
from time import sleep
sleep(0.2)
hc.manual_mode("op")
```

Step 4: Make these changes in the file:

- **Some example files contain at the beginning some unused libraries, these can be removed, for example:**
  import matplotlib.pyplot as plt
  import numpy as npn
  from time import sleep
- **Delete the last few lines, beginning with "hc = LUCIDAC()"**
- **Add these lines at the end:**
  import json
  fobj = open("test.json", "w")
  fobj.write(json.dumps(l.generate())) **where "l" is the name of the previously created "Circuit()" object**
  fobj.close()
- **Save the file as "test.py"**

This is the new file "test.py":

```
# Implement Lorentz attractor as in
# https://analogparadigm.com/downloads/alpaca_2.pdf

from lucipy import LUCIDAC, Circuit
from time import sleep

a = 1.0
b = 2.8
```

```
c = 2.666 / 10

l = Circuit()                           # Create a circuit

mx = l.int(ic = -0.1)
my = l.int(ic = 0.3)
mz = l.int(ic = 0.5)
xz = l.mul()
xy = l.mul()

l.connect(mx, xz.a)                     # Product -x * -z = xz
l.connect(mz, xz.b)

l.connect(mx, xy.a)                     # Product -x * -y = xy
l.connect(my, xy.b)

l.connect(my, mx, weight = -a)
l.connect(mx, mx, weight = a*0.8)       # 0.8 increases stability

l.connect(mx, my, weight = -b)
l.connect(xz, my, weight = -5)
l.connect(my, my, weight = 0.1)

l.connect(xy, mz, weight = 5)
l.connect(mz, mz, weight = c)

l.probe(mx, front_port=0)
l.probe(my, front_port=1)
l.probe(mz, front_port=2)

import json
fobj = open("test.json", "w")
fobj.write(json.dumps(l.generate(), sort_keys=True, indent=4))
fobj.close()
```

Step 5: Type in the command line: python test.py

The file test.json will be created in the same folder:

```
{"/0": {"/U": {"outputs": [0, 2, 0, 1, 1, 0, 0, 8, 1, 9, 2, null, null, null, null, null, null, null, null, null, null, null,
null, 0, 1, 2, null, null, null, null, null]}, "/C": {"elements": [1, 1, 1, 1, -1.0, 0.8, -0.27999999999999997, -0.5, 0.1, 0.5,
0.2666, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0]}, "/I": {"outputs": [[4, 5], [6, 7, 8], [9, 10], [], [], [],
```

```
[], [], [0], [1], [2], [3], [], [], [], []], "upscaling": [false, false, false, false, false, false, true, true, false, true, false,
false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false,
false, false]}, "/M0": {"elements": [{"k": 10000, "ic": -0.1}, {"k": 10000, "ic": 0.3}, {"k": 10000, "ic": 0.5}, {"k": 10000, "ic":
0.0}, {"k": 10000, "ic": 0.0}, {"k": 10000, "ic": 0.0}, {"k": 10000, "ic": 0.0}, {"k": 10000, "ic": 0.0}]}, "/M1": {}}}
```


Step 6 (optional): If you replace the line

fobj.write(json.dumps(l.generate()))

by

fobj.write(json.dumps(l.generate(), indent=4))

then the JSON file contains linefeeds and nice indents:


```
{
    "/0": {
        "/U": {
            "outputs": [
                0,
                2,
                0,
                1,
                1,
                0,
                0,
                8,
                1,
                9,
                2,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                0,
                1,
```

```
            2,
            null,
            null,
            null,
            null,
            null
        ]
    },
    "/C": {
        "elements": [
            1,
            1,
            1,
            1,
            -1.0,
            0.8,
            -0.27999999999999997,
            -0.5,
            0.1,
            0.5,
            0.2666,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            1,
            1,
            1,
            0,
            0,
            0,
            0,
            0
        ]
```

```
        },
        "/I": {
            "outputs": [
                [
                    4,
                    5
                ],
                [
                    6,
                    7,
                    8
                ],
                [
                    9,
                    10
                ],
                [],
                [],
                [],
                [],
                [],
                [
                    0
                ],
                [
                    1
                ],
                [
                    2
                ],
                [
                    3
                ],
                [],
                [],
                [],
                []
            ],
            "upscaling": [
                false,
                false,
                false,
                false,
```

```
            false,
            false,
            true,
            true,
            false,
            true,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false,
            false
        ]
    },
    "/M0": {
        "elements": [
            {
                "k": 10000,
                "ic": -0.1
            },
            {
                "k": 10000,
                "ic": 0.3
            },
            {
                "k": 10000,
                "ic": 0.5
```

```
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                }
            ]
        },
        "/M1": {}
    }
}
```

9.2 Trying to understand the JSON file format

At the time of writing, there exist at least two different versions of the JSON data format. The version created by the online editor is outdated!

Let's compare a few *.py files with the corresponding JSON files:

ttss.py

```
#
# This example of a three-time-scale system is due to Christian Kuehn,
# "Multiple Time Scale Dynamics", Springer, 2015, p. 418., see
# https://analogparadigm.com/downloads/alpaca_44.pdf for details.
#

from lucipy import Circuit, Simulation, LUCIDAC
import matplotlib.pyplot as plt     this line can be removed
import numpy as np                  this line can be removed

c1      = 0.5
c2      = 0.4
c3      = 0.5
epsilon = 0.1
mu      = 0.04

ttss = Circuit()

mx    = ttss.int(ic = 1)
my    = ttss.int()
mz    = ttss.int(slow = True)               # Set k_0 = 100 (default is 10^4)
xx    = ttss.mul()
mxxx  = ttss.mul()
const = ttss.const(1)

ttss.connect(my, mx)
ttss.connect(xx, mx, weight = 10 * c2)        this is > 1
ttss.connect(mxxx, mx, weight = 10 * c3)      this is > 1

ttss.connect(mx, my, weight = -epsilon)     # y' = epsilon(x - z)
ttss.connect(mz, my, weight = epsilon)
```

```
{
    "/0": {
        "/U": {
            "outputs": [
                1,
                8,
                9,
                0,
                2,
                1,
                0,
                15,       constant comes from row 15
                0,
                8,
                0,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                0,        external output on lane 24?
                1,        external output on lane 25?
                2,        external output on lane 26?
```

```
# z' = epsilon^2(mu - c1 y), the factor 100 results from k_0 = 100 instead of
# 10^4. Effectively the 100 cancels out with epsilon * epsilon, but this
# notation makes it possible to vary mu and epsilon without having to rescale
# the z integrator.

ttss.route(const, 7, epsilon * epsilon * mu * 100, mz) (uin, lane, coeff, iout)


ttss.connect(my, mz, weight = epsilon * epsilon * c1 * 100)

ttss.connect(mx, xx.a)                      # xx = x^2
ttss.connect(mx, xx.b)

ttss.connect(xx, mxxx.a)                    # mxxx = -x^3
ttss.connect(mx, mxxx.b)

ttss.probe(mx, front_port=0)       3 external outputs
ttss.probe(my, front_port=1)
ttss.probe(mz, front_port=2)

hc = LUCIDAC()                     this line can be removed

hc.set_circuit(ttss)               this line can be removed

hc.manual_mode("ic")               this line can be removed
from time import sleep             this line can be removed
sleep(0.2)                         this line can be removed
hc.manual_mode("op")               this line can be removed
```

```
                null,
                null,
                null,
                null,
                null
            ],
            "constant": true      l.use_constant(-1)
                                  would generate
                                  "constant": -1
                                  instead

                                  constant(0.1)
                                  would generate
                                  "constant": 0.1
                                  instead
        },
        "/C": {
            "elements": [
                1,
                0.4,
                0.5,
                -0.1,
                0.1,
                0.500000,
                1,
                0.040000,  lane 8, coeff for constant
                1,
                1,
                1,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                1,     output signal has gain 1
```

```
                1,      output signal has gain 1
                1,      output signal has gain 1
                0,
                0,
                0,
                0,
                0
            ]
        },
        "/I": {
            "outputs": [
                [
                    0,
                    1,
                    2
                ],
                [
                    3,
                    4
                ],
                [
                    5,
                    7
                ],
                [],
                [],
                [],
                [],
                [],
                [
                    6
                ],
                [
                    8
                ],
                [
                    9
                ],
                [
                    10
                ],
                [],
                [],
```

```
                [],
                []
            ],
            "upscaling": [
                false,
                true,
                true,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false
            ]
        },
        "/M0": {
            "elements": [
                {
                    "k": 10000,
                    "ic": 1
```

```
                },
                {
                    "k": 10000,
                    "ic": 0
                },
                {
                    "k": 100,
                    "ic": 0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                }
            ]
        },
        "/M1": {}
    }
}
```

Test program with two external inputs, which are multiplied and the product goes to an external output:

```
# Test circuit for external inputs

from lucipy import LUCIDAC, Circuit

test = Circuit()                          # Create a circuit

xy = test.mul()

fp0 = test.front_panel(0) # front panel input/output 0
fp1 = test.front_panel(1) # front panel input/output 1

test.connect(fp0, xy.a)                   # Product fp0 * fp1 = xy
test.connect(fp1, xy.b)

test.probe(xy, front_port=0)

import json
fobj = open("test.json", "w")
fobj.write(json.dumps(test.generate(), indent=4))
fobj.close()
```

```
{
    "/0": {
        "/U": {
            "outputs": [
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                8,                  From the multiplier to lane 24
                null,
                null,
                null,
                null,
                null,
                null,
                null
            ]
        },
        "/C": {
            "elements": [
                0,
```

```
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                1,          Coefficient = 1 for the external output
                1,          This coefficient seems unnecessary
                0,
                0,
                0,
                0,
                0,
                0
            ]
        },
        "/I": {
            "outputs": [
                [],
                [],
                [],
                [],
                [],
                [],
                [],
                [],
                [],
```

	<pre> [24 From lane 24 to upper input of multiplier], [25 From lane 25 to lower input of multiplier], [], [], [], [], [], []], "upscaling": [false,</pre>

```
                false,
                false,
                false
            ]
        },
        "/M0": {
            "elements": [
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                },
                {
                    "k": 10000,
                    "ic": 0.0
                }
            ]
        },
        "/M1": {}
    },
```

	"acl_select": ["external", here the two external inputs are enabled "external", "internal", "internal", "internal", "internal", "internal", "internal"] }

Here https://github.com/anabrid/lucidac-firmware/blob/main/docs/texts/protocol.rst
is written that the Teensy firmware uses this JSON library: https://arduinojson.org/

Here https://github.com/anabrid/lucidac-firmware/blob/main/docs/texts/style-guide.rst
is written that also https://jsonlines.org/ is used, see section "examples".

See also: https://github.com/anabrid/lucidac-firmware/blob/main/lib/communication/schema/block.json

See also (I read it, but didn't understand): https://anabrid.dev/docs/lucidac-firmware/sphinx/dirhtml/texts/protocol/

See also (I read it, but didn't understand): https://anabrid.dev/docs/lucidac-firmware/sphinx/dirhtml/lib-communication/

Communication seems very complicated.

9.3 Other Software

Tinycad is a very simple software for drawing schematic diagrams and exporting the schematic as net list: https://www.tinycad.net/

9.4 Other Digitally Configurable Analog Computers

https://saching007.github.io/pubs/dpac.pdf
https://saching007.github.io/dpac.html

https://rclab.de/en/analogrechner/rg21

C:\Users\astro\Downloads\sensors-23-03599.pdf **(not really digitally configurable, the connections are still made by wires)**

https://yipenghuang.com/research/hybrid/

10 DCAC - Let's build our own Digitally Controlled Analog Computer

Let's keep some good design ideas from Lucidac and change some other things. Lucidac costs 8000 EUR and a selfmade computer should cost much less.

Overall topology:	
16 channels in the math block	Let's use 32 channels, but keep the lower 16 channels compatible with Lucidac
16:8 multiplexer and data acquisition	Let's drop this, data can be acquired by an oscilloscope. Data acquisition could be added later to the Teensy.
16 x 32 U block	Let's use a 32x64 block The U block is described in the "/0 --> /U --> outputs" section in the JSON file. Constants are enabled in the "constant" section of the JSON file. Constants can be 0.1 or 1.0 and are located in row 15 (for lanes 0 to 15) and row 14 (for lanes 16 to 31).
32 channels C block	Let's use 64 channels, keeping the lower 32 channels compatible with Lucidac The C block is described in the "/0 --> /C --> elements" section of the JSON file. Coefficients are defined in software in the [-10 ... +10] range, but are [-1 ... +1] in hardware. If required, 10x gain is set in the I block. Allow coefficients in [-1.0240 ... +1.0235] range with 0.0005 resolution or [-10.240 ... +10.235] range with 0.005 resolution.
8 FP out, 8 FP in	8 inputs seems more than enough, but let's keep 8 inputs for compatibility with Lucidac. External inputs and outputs are on lanes 24 to 31. See lucipy / circuits.py line 94. External inputs are enabled in the "/0 --> acl_select" section in the JSON file.
32 x 16 I block with implicit summing	Implicit summing is a very good idea, let's keep it. Make the I block 64x32. The I block is described in the "/0 --> /I --> outputs" section of the JSON file. 10x gain is enabled in the "/0 --> /I --> upscaling" section in the JSON file.
SH block	Let's drop this
Teensy 4.1	Let's keep it, but let's drop Ethernet, USB is sufficient
+-2 V voltage range	Let's drop this and use +-10 V for compatibility with THAT
Math block	Let's use 32 math channels, keeping the lower 16 channels compatible with Lucidac. In the first step, realize the math block by two THATs, so that the math functions can easily be changed. Software-controlled math functions can be built later.

	The time constants of the integrators are described in the "/0 --> /M0 --> elements --> k" section of the JSON file. The initial conditions of the integrators are described in the "/0 --> /M0 --> elements --> ic" section of the JSON file. The other math block is described in the "/0 --> /M1" section of the JSON file, which is empty. Allow these time constants: k=10000 (TC = 0.1ms) 100kΩ / 1nF k=1000 (TC = 1ms) 100kΩ / 10nF k=100 (TC = 10ms) 1MΩ / 10nF k=10 (TC = 100ms) 1MΩ / 100nF k=1 (TC = 1s) 1MΩ / 1µF It seems like a good idea to put several selectable functions in each math channel.
Software in PC:	
Circuit description in JSON file	Very good idea, let's keep this. The version of the JSON file which is shown in the handbook on pages 14-15 is the latest version. Software in Teensy must be compatible with this version of the JSON file. Default values are used for those variables which aren't specified in the JSON file. For example, if the "acl_select" section is missing in the JSON file, then all members of this section have the default value "internal".
Online circuit editor:	Unfortunately the online editor can't be used because it was developed for an older version of the hardware. The JSON file created by the online editor isn't compatible with the current hardware.
Python software (lucidac.py and lucigui.py)	Let's drop lucidac.py. It's much too complicated to understand and modify the source code. I didn't even find the code that reads and interprets a command line, after one hour of searching. My own hardware won't be compatible with lucidac.py. But lucidac.py is usable for making JSON test files. lucigui.py might be usable, once it has been updated to the latest hardware version. Alternatively, a circuit could be designed with TinyCAD and then compiled by a C# program to a JSON file.
Software in Teensy:	
	Let's drop this, it's much too complicated to modify and much easier to write a new firmware. IC/OP/Halt control is not required, at least not in the first step. One of the THATs is the master. The firmware does only read the JSON file and sets crosspoint switches and coefficients.

I2C bus addresses:

TCA 9548A I2C Multiplexer	I2C Address	Chip
0x00	0x20 - 0x27	PCF8575, channels 0 to 7 in math block
	0x60 - 0x67	MCP4728, channels 0 to 7 in math block
0x01	0x20 - 0x27	PCF8575, channels 8 to 15 in math block
	0x60 - 0x67	MCP4728, channels 8 to 15 in math block
0x02	0x20 - 0x27	PCF8575, channels 16 to 23 in math block
	0x60 - 0x67	MCP4728, channels 16 to 23 in math block
0x03	0x20 - 0x27	PCF8575, channels 24 to 31 in math block
	0x60 - 0x67	MCP4728, channels 24 to 31 in math block
0x04	0x20	PCF8575, gain switching for channels 0 to 15 in C block
	0x21	PCF8575, gain switching for channels 16 to 31 in C block
	0x22	PCF8575, gain switching for channels 32 to 47 in C block
	0x23	PCF8575, gain switching for channels 48 to 63 in C block
	0x24	PCF8575, 8 switches for external inputs in Cblock
	0x25	PCF8575, 2 switches for constant inserter in U block
	0x27	LCD Display with 40x4 characters
0x05		reserved
0x06		reserved
0x07		reserved
don't care	0x70	TCA9548A I2C Multiplexer

The coefficients in the C block are controlled by a SPI port daisy chain.

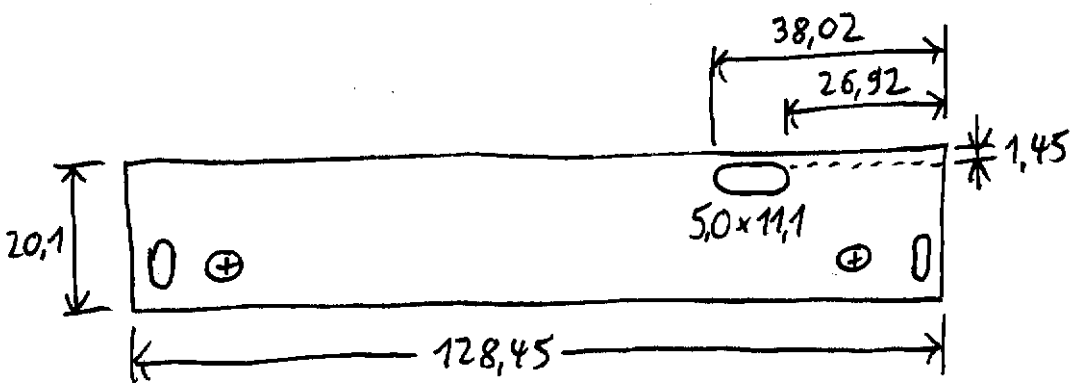
The crosspoint switches in the U and I blocks are controlled by 3 pins from the microcontroller.

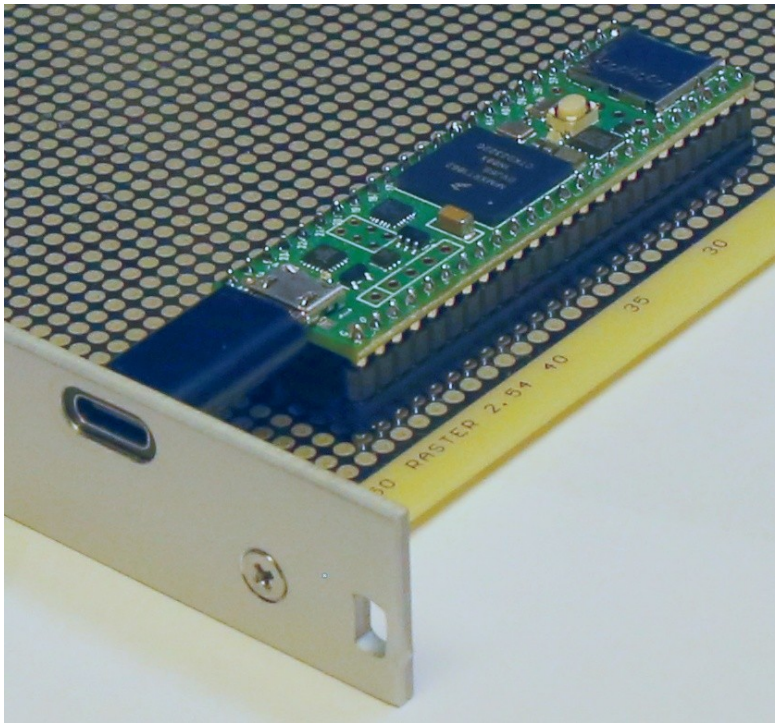
10.1 What's on which board?

The idea is to use a standard 19" housing.

Module, from left to right	Width units, total 84	What's on the board?	Connector
Power supply	8	Switching power supply +-15 V, 75 W	
Controller + Capture	4	Teensy 4.1, 2 x AD75019, MCP3304	96 pin
U Block	4	32 x Zener protectors + overload detectors, 8 x AD75019, +-12 V regulators, TL082	160 pin
C Block (0 - 15) with gain switching	4	8 x AD5449, 8 x TL084, 4 x DG212, 1 x PCF8575	96 pin
C Block (16 - 31) with gain switching and with external I/O	4	8 x AD5449, 8 x TL084, 4 x DG212, 2 x PCF8575, 2 x DG213, 4 x TL084 (4 x TL084 and Zener protectors for external I/O can be placed on another board near the connectors)	96 pin
C Block (32 - 47) with gain switching	4	8 x AD5449, 8 x TL084, 4 x DG212, 1 x PCF8575	96 pin
C Block (48 - 63) with gain switching	4	8 x AD5449, 8 x TL084, 4 x DG212, 1 x PCF8575	96 pin
I Block	4	8 x AD75019, +-12 V regulators, 8 x TL084	160 pin
Math Block (0 - 3)	4	Integrators with limiters	96 pin
Math Block (4 - 7)	4	Integrators with limiters	96 pin
Math Block (8 - 15), constant inserter	4	4 Multipliers, 4 x identity or ext. functions (2mm connectors), 2 x DG213, 1 x TL082, 1 x PCF8575	96 pin
Math Block (16 - 19) a	4	Integrators with limiters / Multipling_Integrators / Solvers	96 pin
Math Block (16 - 19) b	4	Multiply / Divide / Square / Square root / Open amplifier	96 pin
Math Block (20 - 23) a	4	Integrators with limiters	96 pin
Math Block (20 - 23) b	4	Analog power series	96 pin
Math Block (24 - 27) a	4	Analog exponential function and logarithm	96 pin
Math Block (24 - 27) b	4	Analog sin/cos/tan/atan	96 pin
Math Block (28 - 31) a	4	Comparators	96 pin
Math Block (28 - 31) b	4	Digital functions (lookup tables, wide range arctan, optimizers), Teensy 4.0	96 pin
Spare	4		

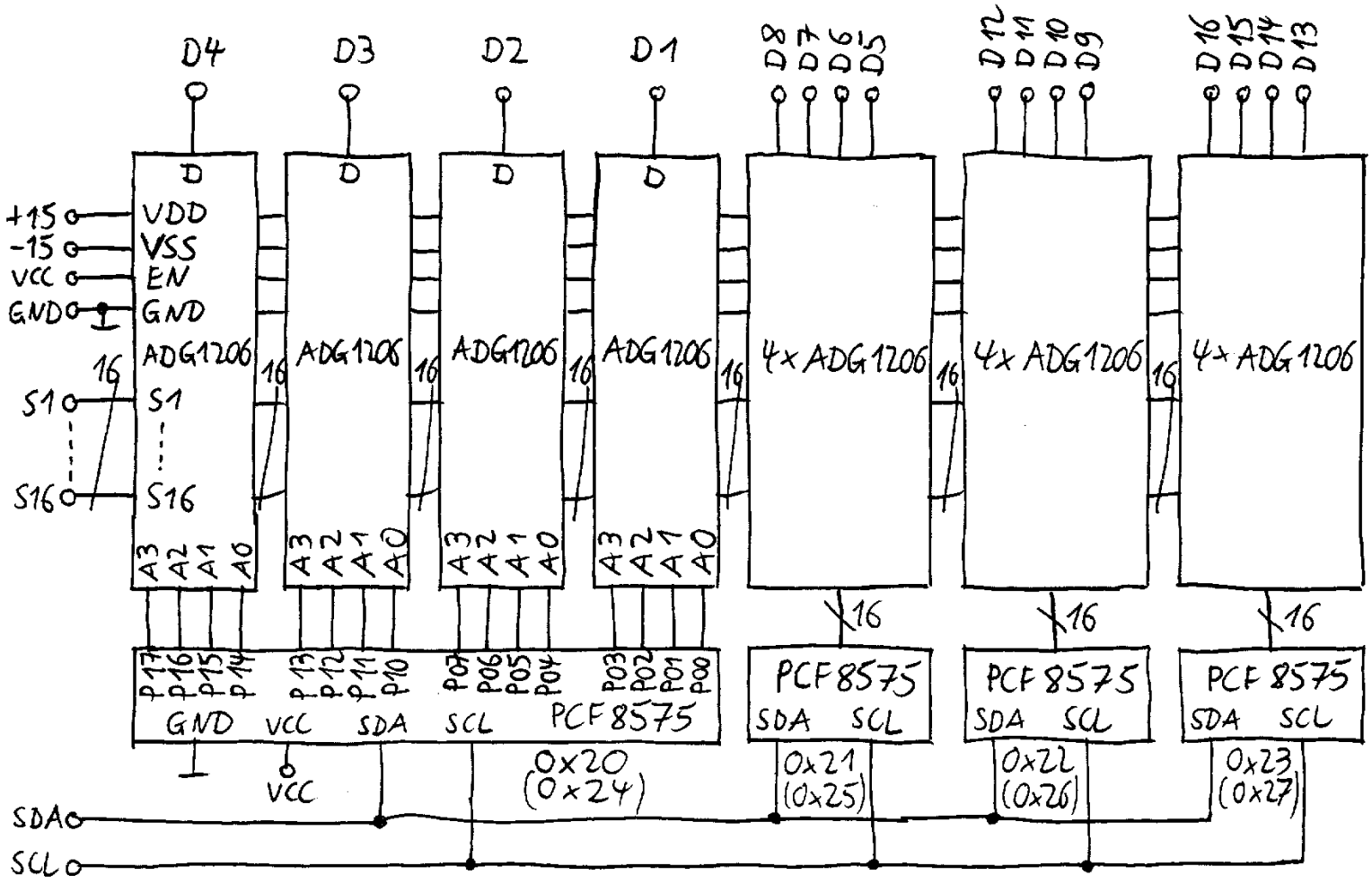
10.2 Controller + Capture Board

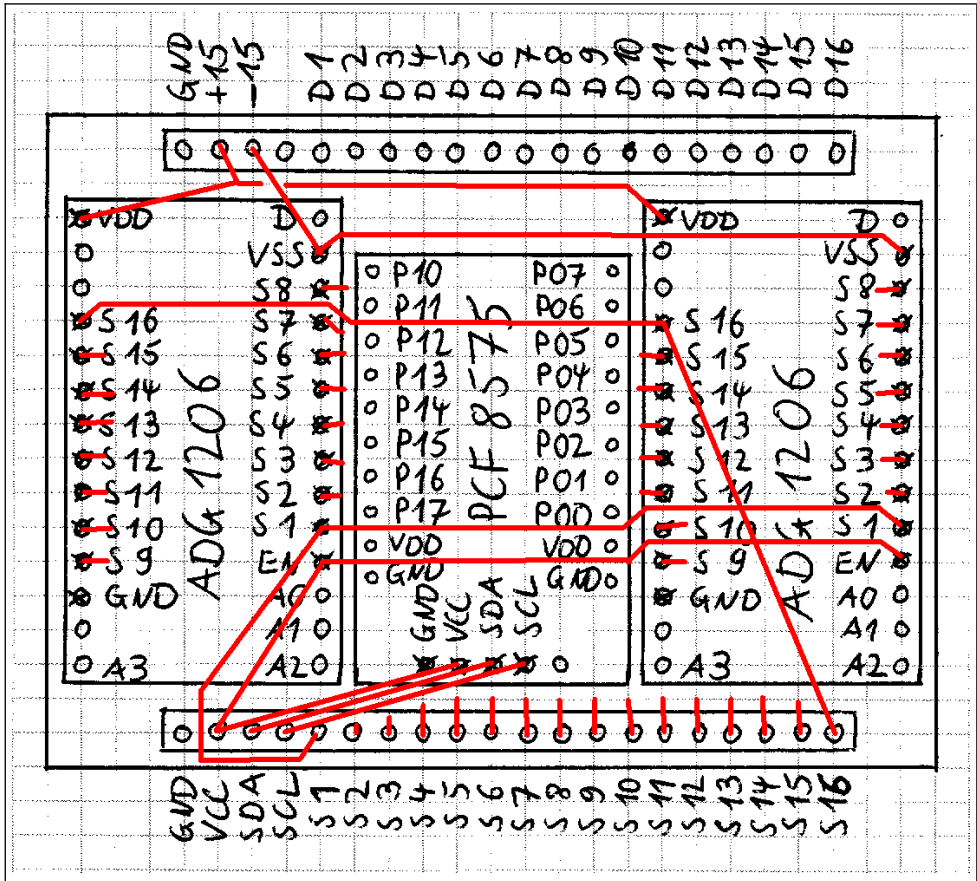




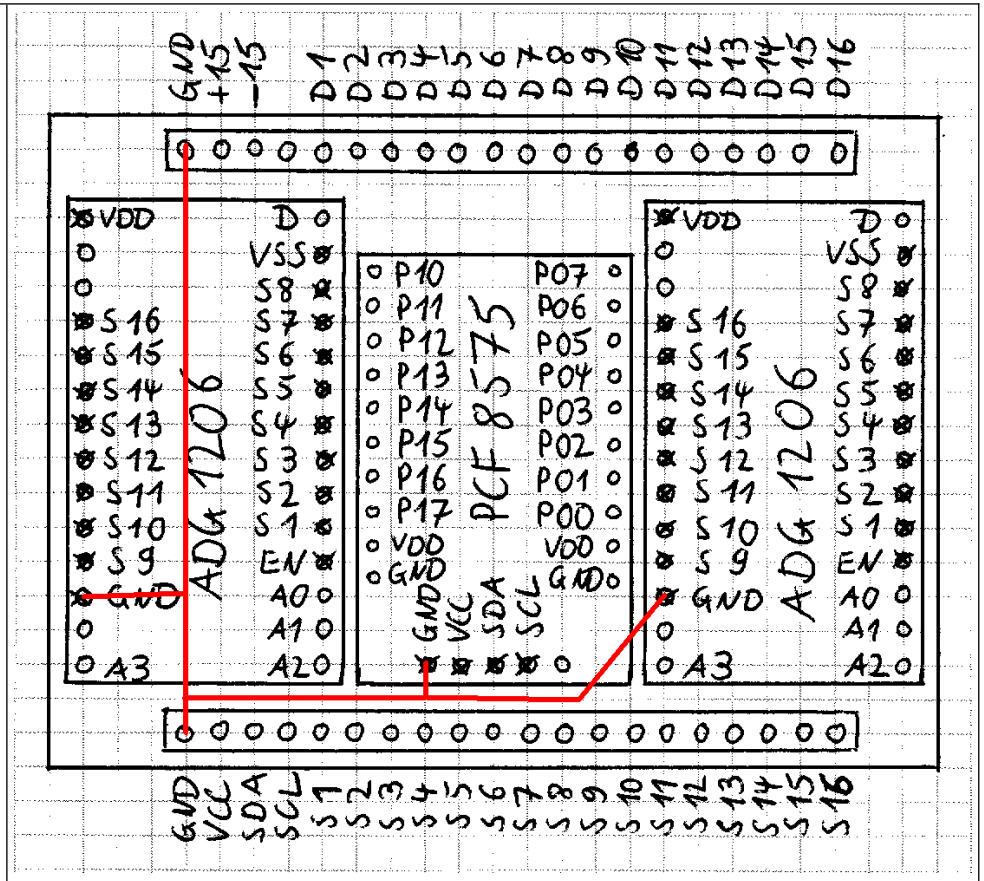
10.3 16:16 Multiplexer Module for +-10V signals (outdated)

Schmematic diagram:

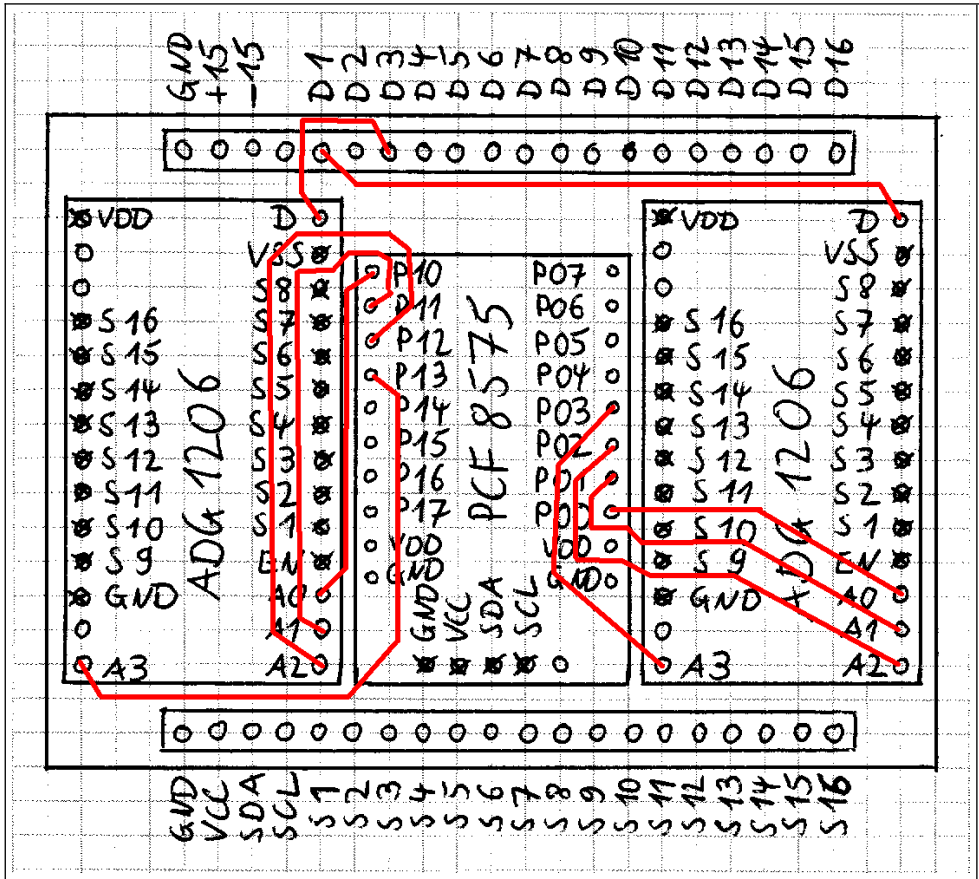




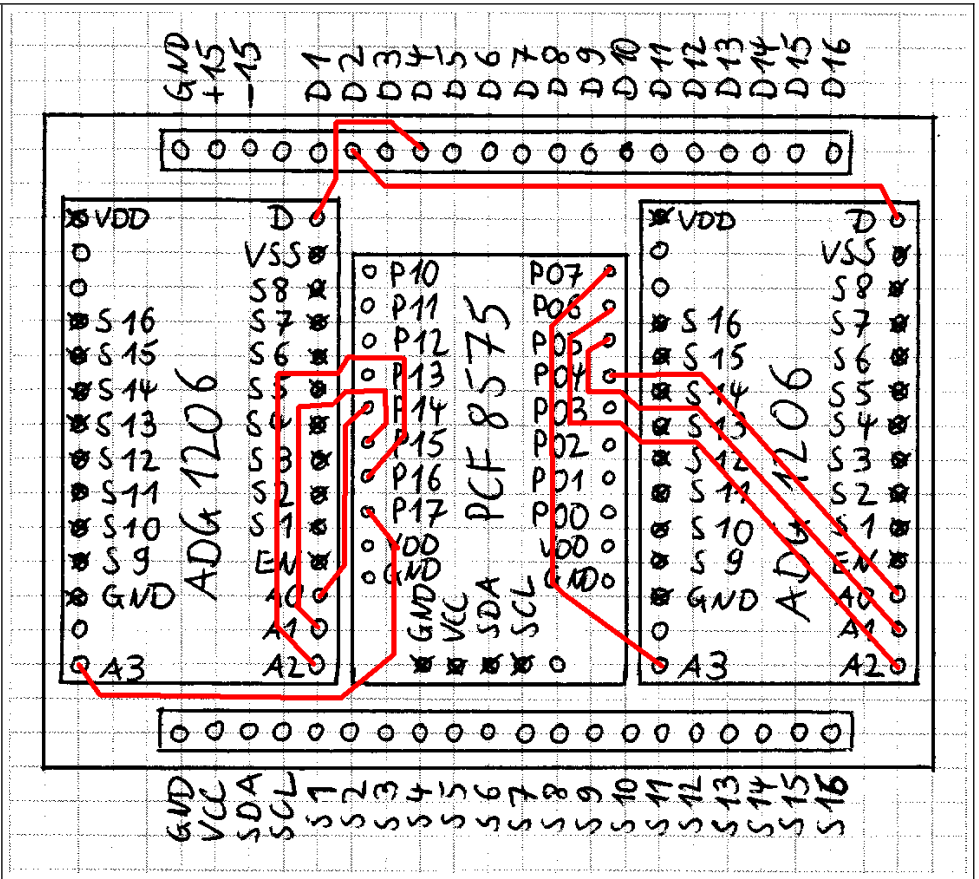
Wiring on bottom layer of base board, including all S1 to S16 signals
Pins shown with "X" are connected to all stacked modules.



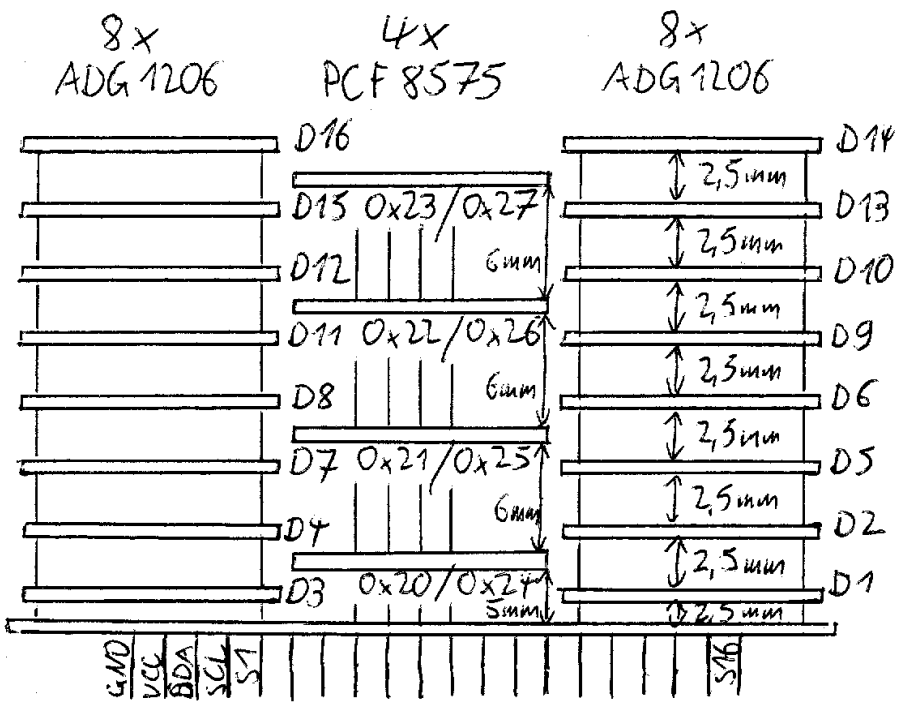
Wiring on top layer of base board (only GND).



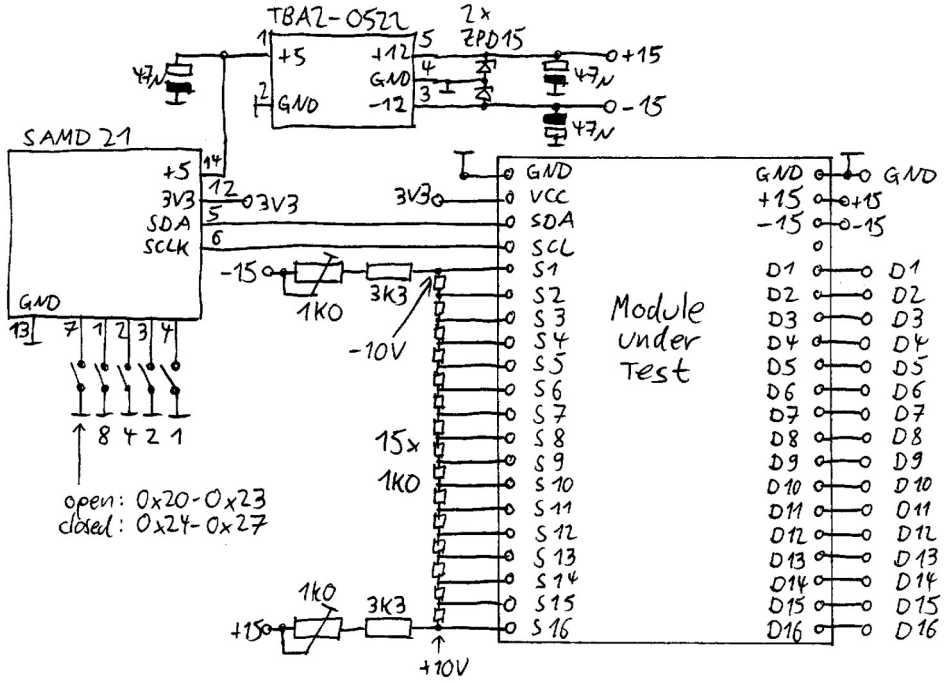
Wiring of the first ADG1206 layer and first PCF8575 layer.
Connect VCC to VDD on the bottom side of all PCF8575 modules.
Set the I2C address on the bottom side of all PCF8575 modules.



Wiring of the second ADG1206 layer and first PCF8575 layer.
Connect VCC to VDD on the bottom side of all PCF8575 modules.
Set the I2C address on the bottom side of all PCF8575 modules.



Side view of the module, seen from the side where the S1 to S16 pins are.

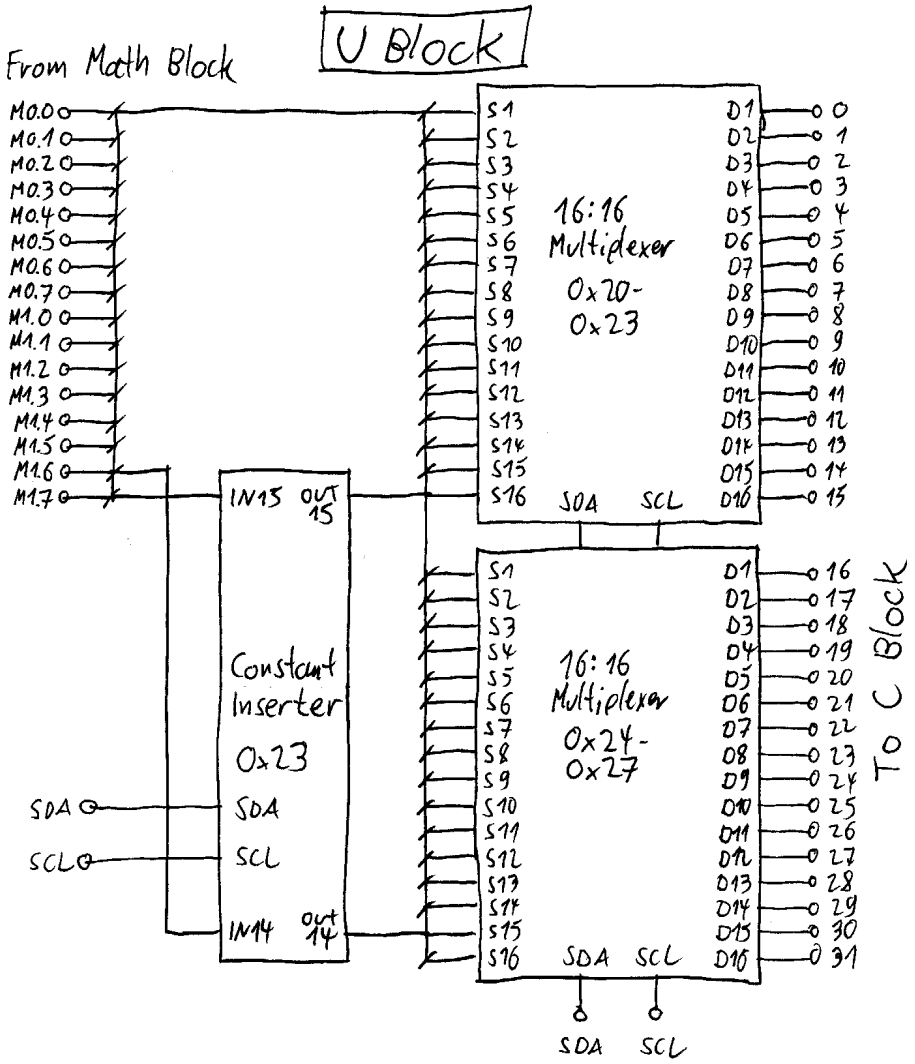


Test circuit for the 16:16 multiplexer module. It's important to make a test each time before the next component is soldered on top of the stack, because the components below aren't accessible any more.

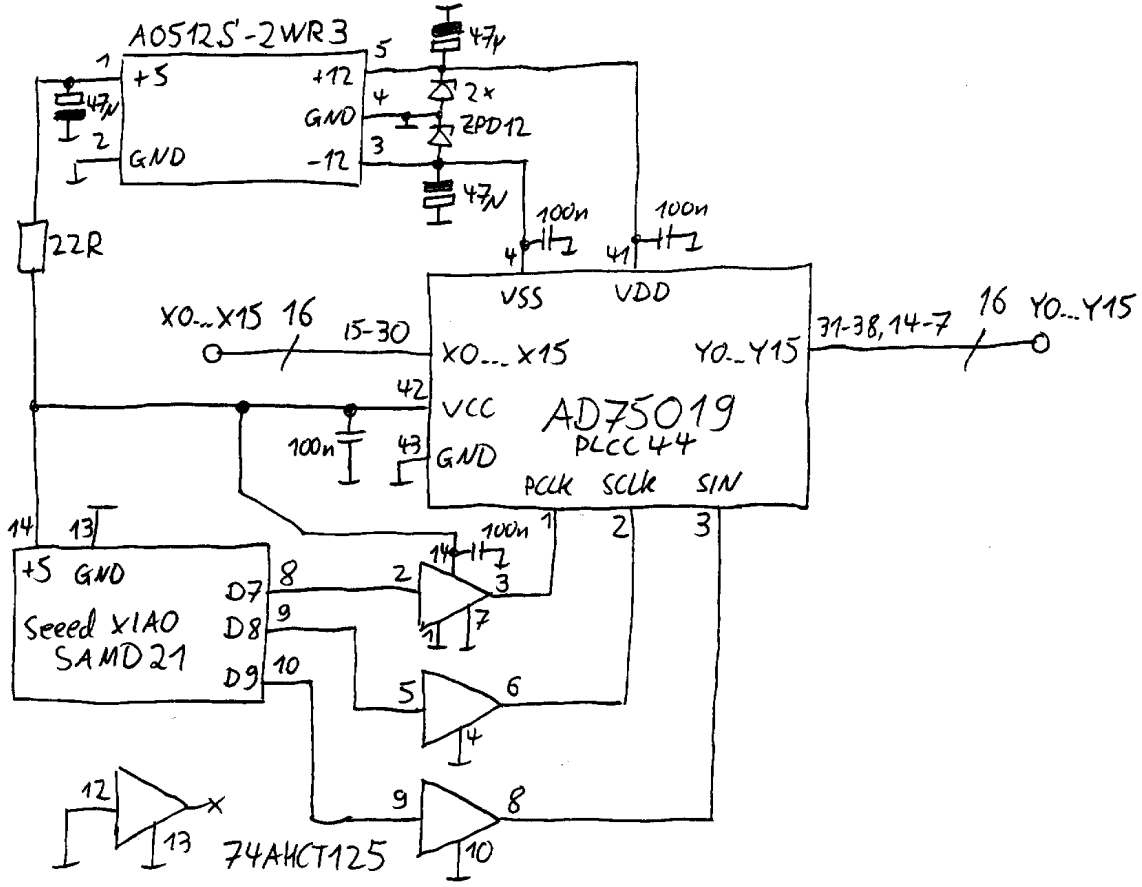
Software for the test circuit:

```
#include <Wire.h>

int d, addr, b, x;

void setup()
{
  pinMode(D0, INPUT_PULLUP); // pullup resistors for mode switch inputs
  pinMode(D1, INPUT_PULLUP);
  pinMode(D2, INPUT_PULLUP);
  pinMode(D3, INPUT_PULLUP);
  pinMode(D6, INPUT_PULLUP);
  Wire.begin();
}

void loop()
{
  d = 15 - digitalRead(D3);
  d -= 2 * digitalRead(D2);
  d -= 4 * digitalRead(D1);
  d -= 8 * digitalRead(D0);

  addr = 0x20 + (d / 4);  // from 0x20 to 0x23
  if (!digitalRead(D6))
    addr += 4;            // from 0x24 to 0x27

  b = 1 << (4 * (d%4));   // 1, 16, 256 or 4096

  for (int i = 0; i < 16; i++)
  {
    x = i * b;
    Wire.beginTransmission(addr);
    Wire.write(lowByte(x));
    Wire.write(highByte(x));
    Wire.endTransmission();
    delay(5);
    if (i == 0)
      delay(10);
  }
  delay(10);
}
```

10.4 U Block (outdated)



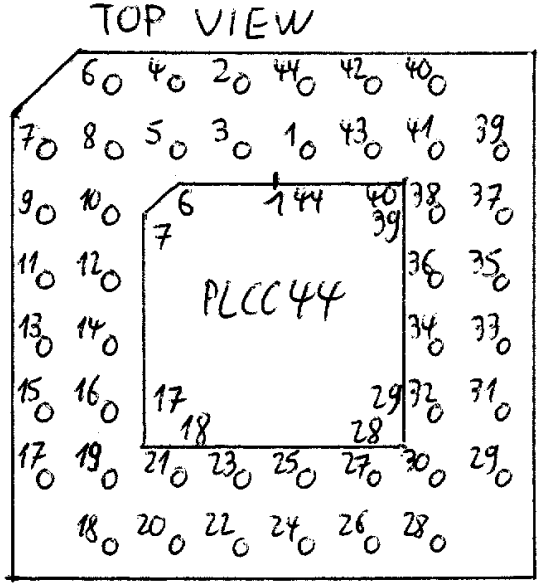
10.5 Test Circuit for AD75019 Crosspoint Switch

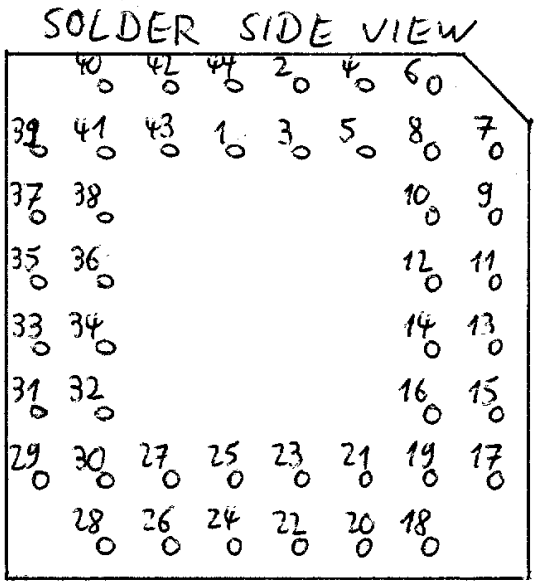


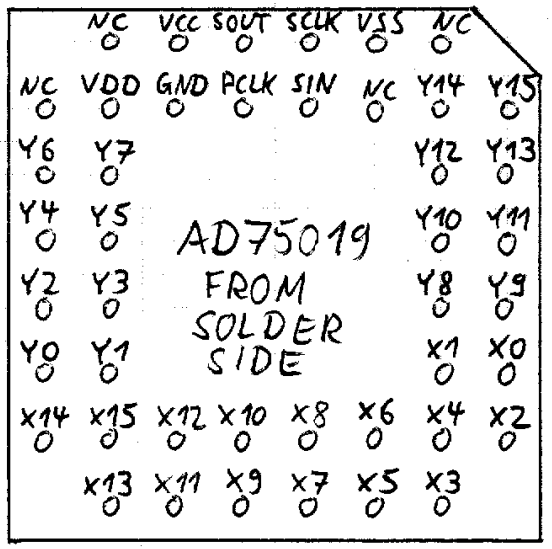
The 3.3V to 5V level conversion with 74AHCT125 is unnecessary, because the AD75019 accepts 2.4V as a logic high.

Add a 1N4001 diode between VCC (anode) and VDD (cathode), to make sure that VDD can't become smaller than VCC. This is described in the datasheet.

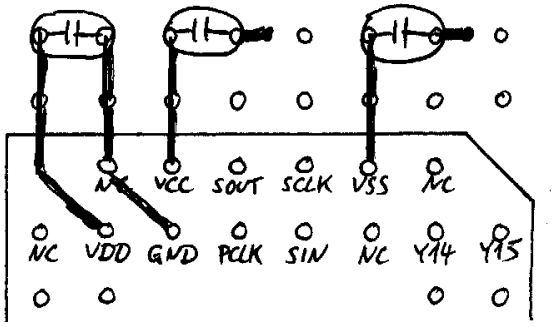
Datasheet 74AHCT125: https://assets.nexperia.com/documents/data-sheet/74AHC_AHCT125.pdf







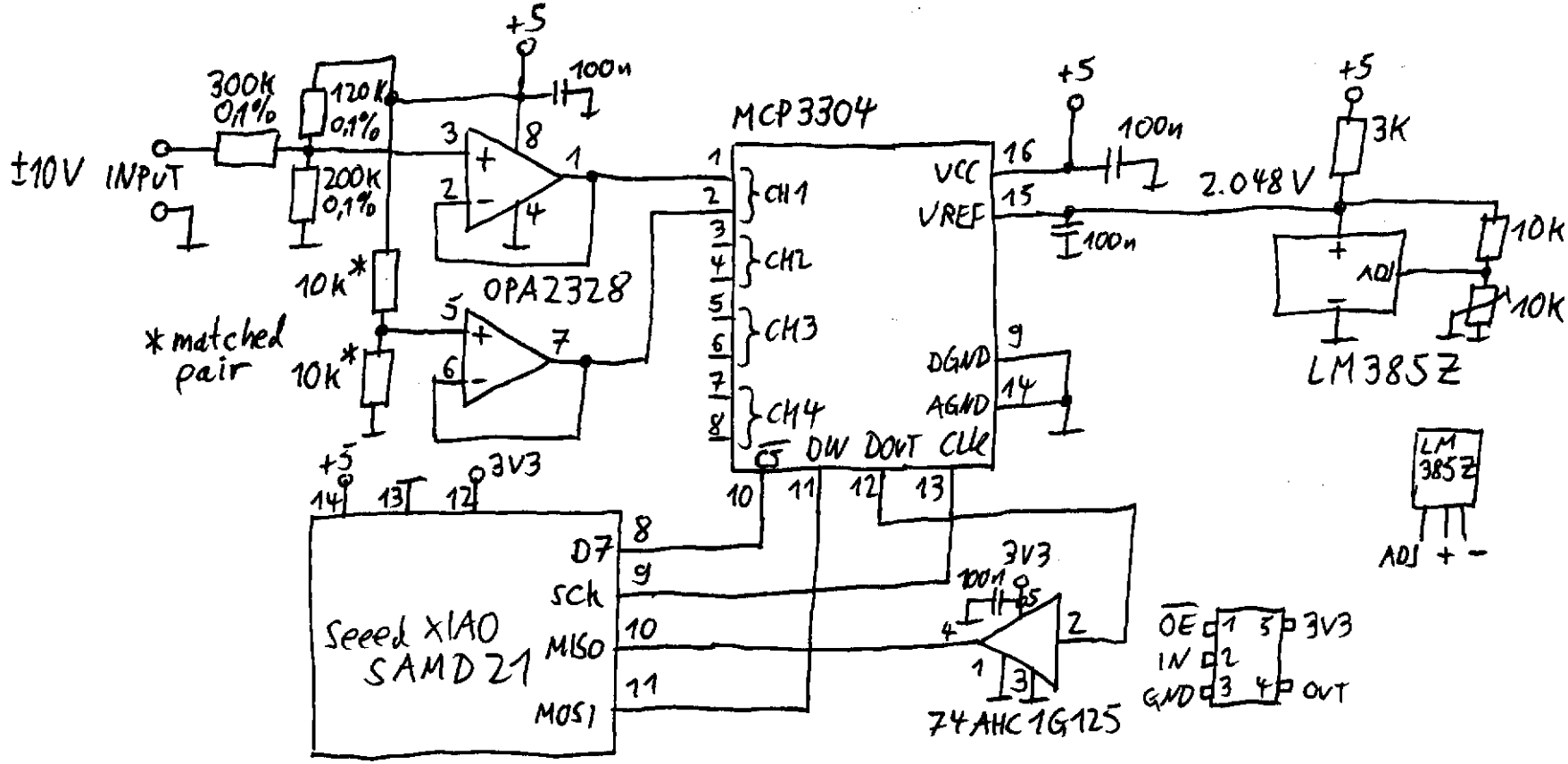
Blocking capacitors, seen from solder side:



Software for testing:

```
int x, y;

void setup()
{
  pinMode(D7, OUTPUT);   // PCLK
  pinMode(D8, OUTPUT);   // SCLK
  pinMode(D9, OUTPUT);   // SIN

  digitalWrite(D7,1);
  digitalWrite(D8,0);
}

void loop()
{
  noInterrupts();
  digitalWrite(D7,0);    // PCLK = 0
  delayMicroseconds(1);
  digitalWrite(D7,1);    // PCLK = 1
  delayMicroseconds(5);
  for(y = 15; y >= 0; y--)
  {
    for(x = 15; x >= 0; x--)
    {
      if((x == 7) && (y == 15))
        digitalWrite(D9,1);   // SIN = 1   the X7/Y15 switch is closed
      else
        digitalWrite(D9,0);   // SIN = 0   all other switches are open
      delayMicroseconds(1);
      digitalWrite(D8,1);    // SCLK = 1
      delayMicroseconds(1);
      digitalWrite(D8,0);    // SCLK = 0
    }
  }
  interrupts();
  delay(5);
}
```

10.6 Analog to Digital Conversion with MCP3304



Gain = 0.2 120kΩ || 200kΩ = 75kΩ 300kΩ || 200kΩ = 120kΩ

The resolution of the +-10 V input signal is exactly 2.5 mV, or 0.00025 machine units.

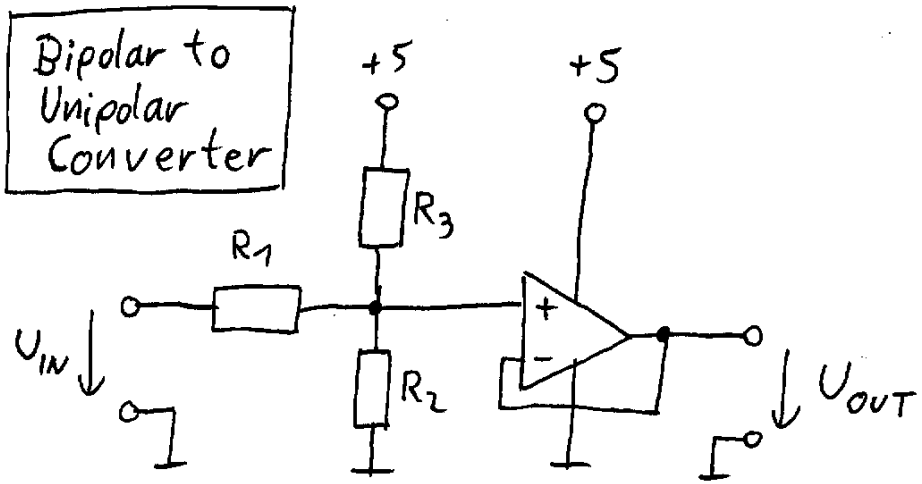
Possible 5V rail-to-rail OPs:

OP-Amp	Channels per Chip	Maximum Offset Voltage	Price per Chip (Mouser, 10 pcs)
AD8628	1	5 µV	2.47 EUR
AD8629	2	5 µV	3.81 EUR
AD8630	4	5 µV	6.81 EUR
MCP606	1	250 µV	0.70 EUR
MCP607	2	250 µV	1.22 EUR
MCP609	4	250 µV	1.50 EUR
MCP6021	1	500 µV	1.05 EUR
MCP6022	2	500 µV	1.41 EUR
MCP6024	4	500 µV	2.06 EUR
OPA328	1	50 µV	2.41 EUR
OPA2328	2	50 µV	2.34 EUR
OPA387	1	2 µV	1.42 EUR
OPA2387	2	2 µV	2.28 EUR
OPA4387	4	2 µV	3.04 EUR
OPA397	1	60 µV	1.14 EUR

Possible combinations of resistors:

R1	R2	R3
300 kΩ	200 kΩ	120 kΩ
600 kΩ	400 kΩ	240 kΩ
750 kΩ	500 kΩ	300 kΩ
900 kΩ	600 kΩ	360 kΩ
1.5 MΩ	1 MΩ	600 kΩ
1.8 MΩ	1.2 MΩ	720 kΩ

Non-inverting bipolar to unipolar converter, see also:
https://www.ti.com/lit/an/sbaa604/sbaa604.pdf?ts=1735188085085
See also: https://www.electronicdeveloper.de/ADCRangeAktiv.aspx



$$Gain = \frac{\Delta U_{OUT}}{\Delta U_{IN}} \qquad Gain < 0{,}5$$

$$R_1 = R_2 \left(\frac{1}{2 \cdot Gain} - 1 \right)$$

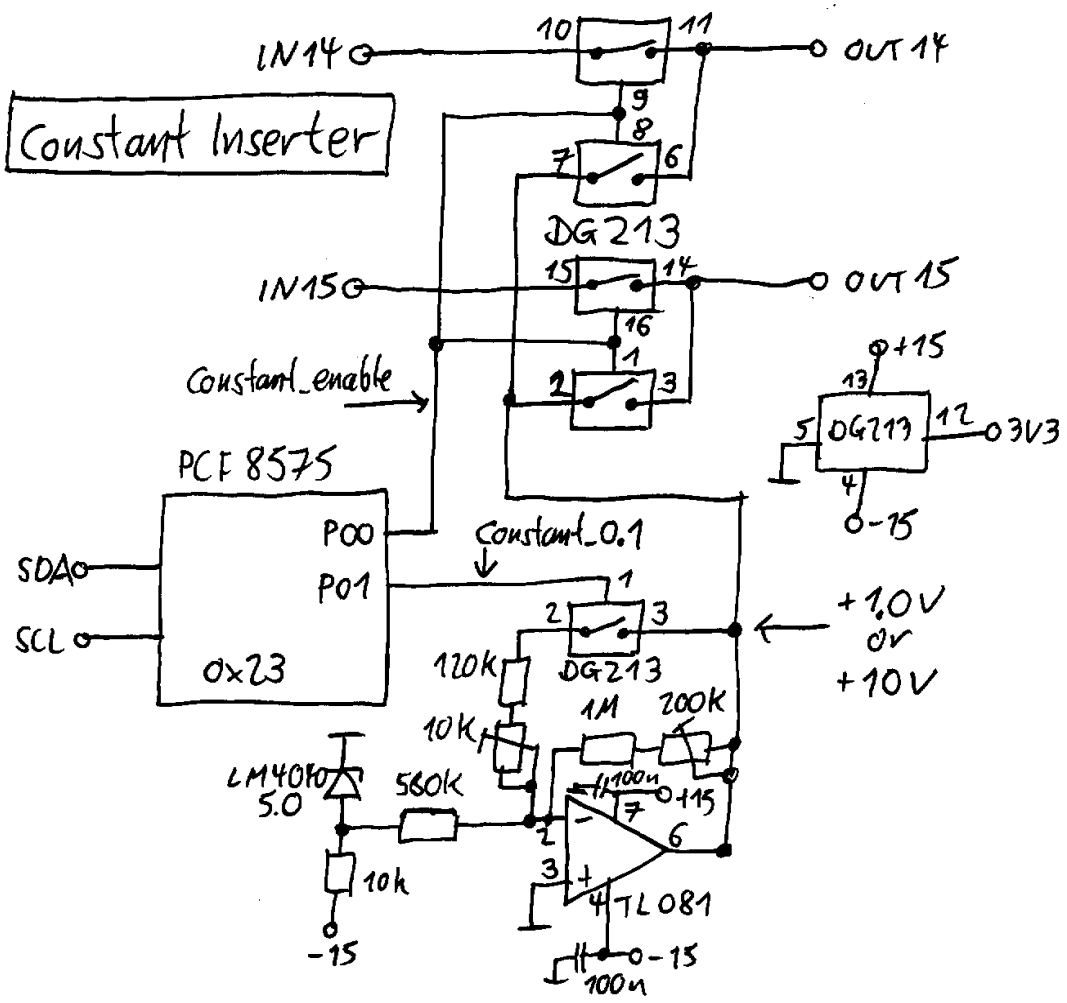
$$R_3 = R_2 \left(1 - 2 \cdot Gain \right)$$

$$R_1 \,||\, R_2 = R_3 \qquad R_2 \,||\, R_3 = R_1 \cdot \frac{Gain}{1 - Gain}$$

Test software for MCP3304:

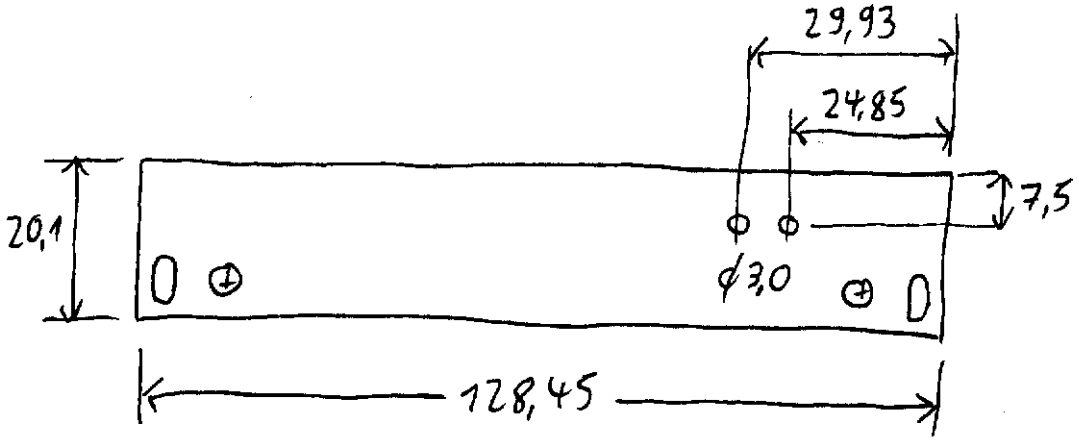
```
/*    Test for MCP3304 ADC    Michael Koch, January 2025  */

#include <SPI.h>

void setup()
{
  pinMode(D7,OUTPUT);       //  /CS
  pinMode(D8,OUTPUT);       //  SCK
  pinMode(D9,INPUT);        //  D_OUT
  pinMode(D10,OUTPUT);      //  D_IN

  SPI.begin ();
  SPI.setBitOrder (MSBFIRST);
  SPI.setDataMode (SPI_MODE0);
  SPI.setClockDivider (SPI_CLOCK_DIV16);
  digitalWrite(D7,HIGH);    //  /CS = high
}

void loop()
{
  digitalWrite(D7,LOW);    // /CS = low
  SPI.transfer(0x08);      // write: 0 0 0 0 1 0 D2 D1
                           //  0x08 for channels 0,1
                           //  0x09 for channels 2,3
                           //  0x0A for channels 4,5
                           //  0x0B for channels 6,7
  int adcValue = SPI.transfer16(0x0000);  // write: D0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                                          // read: x x 0 sign 12_databits
  digitalWrite(D7,HIGH);  // /CS = high
      if(adcValue & 0x1000)   // Sign Bit
    adcValue = (adcValue & 0x0fff) - 0x1000;
  else
    adcValue &= 0x0fff;

  Serial.println((float)adcValue / 4000, 5);   // Resolution is 1/4000 = 0.00025
  delay(500);
}
```

10.7 Constant Inserter



Replace the second DG213 by DG419, which has a DIP8 package.

10.8 U Block



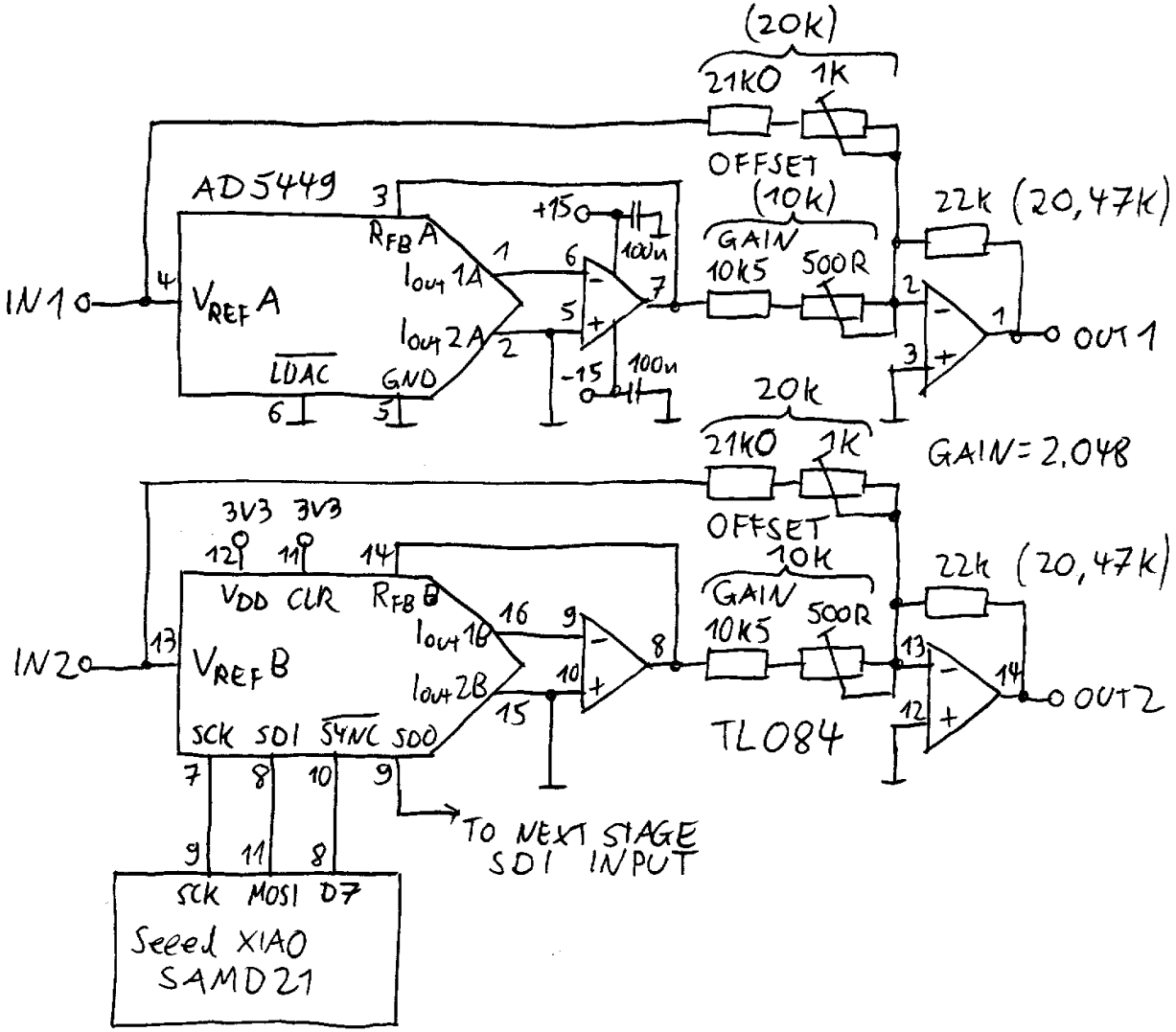
10.9 C Block

Repeat this block 16 times. The SPI port is daisy-chained through all AD5449 chips.

Constants can be set in the [-1.0240 ... +1.0235] range with 0.0005 resolution, or in the [-10.240 ... +10.235] range with 0.005 resolution.

The 21k5 resistor must be changed to 21k0.
The exact resistor values (for range -1.0235 to +1.0235) are:
22k feedback
10.742k gain (11k / 1.0235)
21.484k offset (22k / 1.0235)

Alternatively, the trimmers can be omitted if 0.1% resistors are used:
20.47k feedback (20k · 1.0235) = 20k + 470R
10k gain
20k offset



Test software for Seeed XIAO SAMD21:

```
/*    Test for AD5449 Multiplying DAC    Michael Koch, October 2024  */

#include <SPI.h>

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(D7,OUTPUT);      // connect to SYNC
  pinMode(D8,OUTPUT);      // connect to SCK
  pinMode(D10,OUTPUT);     // connect to SDI
  digitalWrite(D7,HIGH);   // SYNC = high

  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV2);  // divide the clock by 2
  SPI.setDataMode(SPI_MODE1);
}

void loop()
{
  int v = 0x800 - 2000;         // Gain = -1.0000
  digitalWrite(D7,LOW);         // SYNC = low
  SPI.transfer16(0x1000 + v);   // Channel 1
  digitalWrite(D7,HIGH);        // SYNC = high
  delayMicroseconds(1);
  digitalWrite(D7,LOW);         // SYNC = low
  SPI.transfer16(0x4000 + v);   // Channel 2
  digitalWrite(D7,HIGH);        // SYNC = high
  delay(2500);
  v = 0x800 + 2000;             // Gain = +1.0000
  digitalWrite(D7,LOW);         // SYNC = low
  SPI.transfer16(0x1000 + v);   // Channel 1
  digitalWrite(D7,HIGH);        // SYNC = high
  delayMicroseconds(1);
  digitalWrite(D7,LOW);         // SYNC = low
  SPI.transfer16(0x4000 + v);   // Channel 2
  digitalWrite(D7,HIGH);        // SYNC = high
  delay(2500);
}
```

10.10 I Block (outdated)

This block contains:

- **8 external outputs**
- **8 switchable external inputs**
- **Gain switching 1x / 10x for all 32 channels**
- **Implicit summing in the I block**

To reduce the error due to the R_{ON} of the DG212 analog switches, it might be better to multiply all resistors by a factor 10:
900k, 100k, 1M

Assuming R_{ON} = 100Ω, these resistors could be used:
91k, 10k, 101k (=100k + 1k)

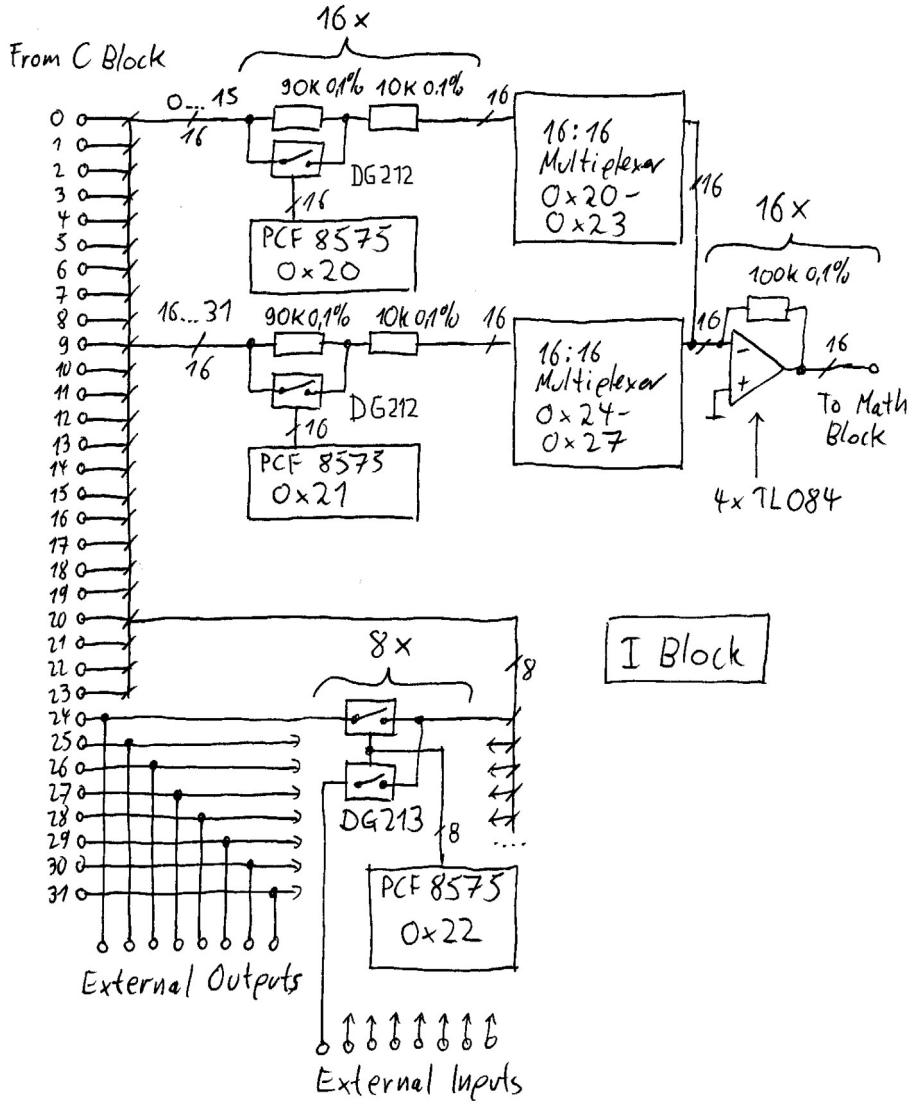
Problem: The I block reverses the sign

Possible solutions:

a) 32 Inverters after the I block, 8 x TL084 required

b) Invert all 64 coefficients in software, and invert the 8 external inputs and 8 external outputs. 4 x TL084 required.

Solution b) is better. It's good to have OP-Amp stages between the external inputs/outputs and the crosspoint switches.



10.11 LED Matrix for Visualisation of the Coefficients

Using this LED matrix: https://de.aliexpress.com/item/1005003901833984.html

Test program for 16x16 matrix of WS2812 LEDs with Teensy 4.0 and 74AHCT1G125 driver:

```
// 16x16 Array RGB-LED Test, for Teensy 4.0
// Michael Koch 2024

int DI = 23;

#define LEDS 256

#define T1H 900
#define T1L 350
#define T0H 350
#define T0L 900

int a[LEDS];

void setup()
{
  pinMode(DI, OUTPUT);  // initialize the digital pin as an output

  a[0] = 0x00002000;     // red
  a[1] = 0x00102000;     // orange
  a[2] = 0x00202000;     // yellow
  a[3] = 0x00200000;     // green
  a[4] = 0x0020000c;     // blue-green
  a[5] = 0x00100020;     // cyan
```

```
  a[6] = 0x00000020;     // blue
  a[7] = 0x00001020;     // violet
}

void clear_leds()
{
  for(int i = 0; i < LEDS; i++)
    a[i] = 0;
}

void set_led(int x, int y, int color)   // 0,0 is top left
{
  a[255 - x - 16 * y + (y%2) * (2 * x - 15)] = color;
}

void write_leds()
{
  digitalWriteFast(DI, LOW);
  delayMicroseconds(70);  // reset     Use at least 70000ns for small 2mm x 2mm RGB LEDs !

  noInterrupts();
  for (int n = 0; n < LEDS; n++)
  {
    int rgb = a[n];
    for (int i = 0; i < 24; i++)
    {
      digitalWriteFast(DI, HIGH);
      if ((rgb <<= 1) & 0x01000000)  // write logic 1
      {
        delayNanoseconds(T1H);
        digitalWriteFast(DI, LOW);
```

```
        delayNanoseconds(T1L);
      }
      else                          // write logic 0
      {
        delayNanoseconds(T0H);
        digitalWriteFast(DI, LOW);
        delayNanoseconds(T0L);
      }
    }
  }
  interrupts();
}

void loop()
{
  clear_leds();
  a[0] = 0x00002000;     // red
  a[1] = 0x00102000;     // orange
  a[2] = 0x00202000;     // yellow
  a[3] = 0x00200000;     // green
  a[4] = 0x0020000c;     // blue-green
  a[5] = 0x00100020;     // cyan
  a[6] = 0x00000020;     // blue
  a[7] = 0x00001020;     // violet

  set_led(0,13,0x00000100);
  set_led(1,13,0x00000200);
  set_led(2,13,0x00000400);
  set_led(3,13,0x00000800);
  set_led(4,13,0x00001000);
  set_led(5,13,0x00002000);
```

```
  set_led(6,13,0x00004000);
  set_led(7,13,0x00008000);
  set_led(8,13,0x0000ff00);

  set_led(0,11,0x00000001);
  set_led(1,11,0x00000002);
  set_led(2,11,0x00000004);
  set_led(3,11,0x00000008);
  set_led(4,11,0x00000010);
  set_led(5,11,0x00000020);
  set_led(6,11,0x00000040);
  set_led(7,11,0x00000080);
  set_led(8,11,0x000000ff);

  for(int i = 0; i < 16; i++)
    set_led(i,i,0x00100000);
  write_leds();
  delay(500);
}
```

Test software for 32x32 LED array:
4 arrays 16x16 each, input is bottom left, then top left, then bottom right, then top right.
For full brightness of all 1024 LEDs, a 5V 40A power supply is required.

```
// 32x32 Array RGB-LED Test, for Teensy 4.0
// Michael Koch 2024

int DI = 23;
int b;

#define LEDS 1024

#define T1H 900
#define T1L 350
#define T0H 350
#define T0L 900

int a[LEDS];

void setup()
{
  pinMode(DI, OUTPUT);  // initialize the digital pin as an output
  digitalWriteFast(DI, LOW);
}

void clear_leds()
{
  for(int i = 0; i < LEDS; i++)
    a[i] = 0;
}

void set_led(int x, int y, int color)   // 0,0 is top left
{
  a[511 + 512 * (x/16) - (x%16) - 16 * y + (y%2) * (2 * (x%16) - 15)] = color;
}

void write_leds()
{
  digitalWriteFast(DI, LOW);
  delayMicroseconds(70);  // reset     Use at least 70000ns for small 2mm x 2mm RGB LEDs !

  noInterrupts();
  for (int n = 0; n < LEDS; n++)
```

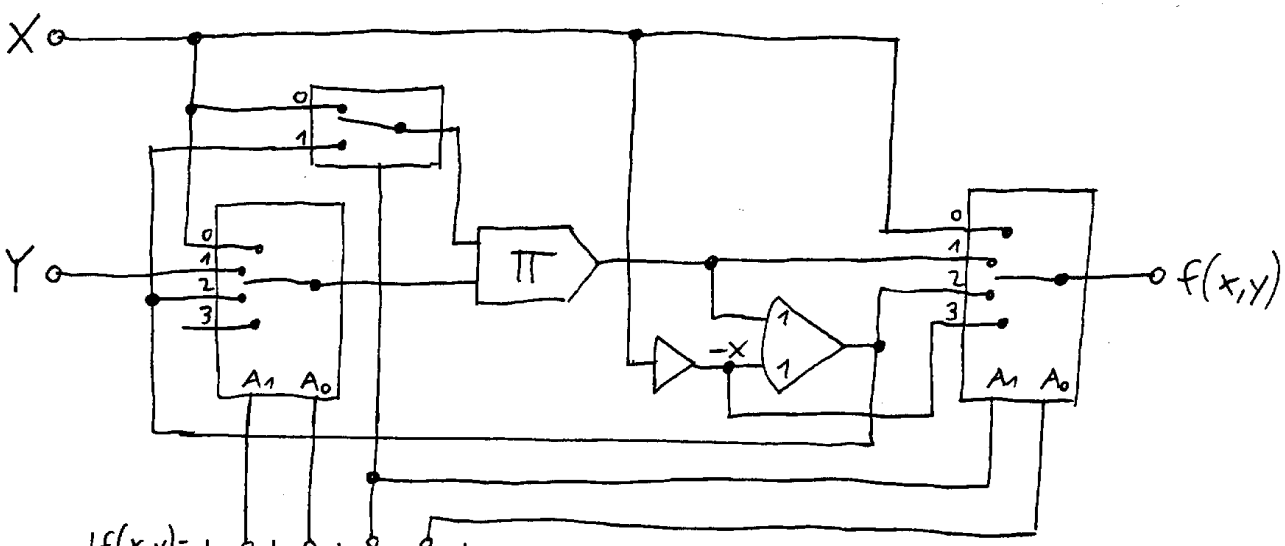
```
  {
    int rgb = a[n];
    for (int i = 0; i < 24; i++)
    {
      digitalWriteFast(DI, HIGH);
      if ((rgb <<= 1) & 0x01000000)  // write logic 1
      {
        delayNanoseconds(T1H);
        digitalWriteFast(DI, LOW);
        delayNanoseconds(T1L);
      }
      else                           // write logic 0
      {
        delayNanoseconds(T0H);
        digitalWriteFast(DI, LOW);
        delayNanoseconds(T0L);
      }
    }
  }
  interrupts();
}

void loop()
{
  clear_leds();

  set_led(0,13,0x00000100);     // blue, different intensities
  set_led(1,13,0x00000200);
  set_led(2,13,0x00000400);
  set_led(3,13,0x00000800);
  set_led(4,13,0x00001000);
  set_led(5,13,0x00002000);
  set_led(6,13,0x00004000);
  set_led(7,13,0x00008000);
  set_led(8,13,0x0000ff00);

  set_led(0,11,0x00000001);     // red, different intensities
  set_led(1,11,0x00000002);
  set_led(2,11,0x00000004);
  set_led(3,11,0x00000008);
  set_led(4,11,0x00000010);
  set_led(5,11,0x00000020);
  set_led(6,11,0x00000040);
```

```
  set_led(7,11,0x00000080);
  set_led(8,11,0x000000ff);

  set_led(0,15,0x00002000);     // red
  set_led(1,15,0x00102000);     // orange
  set_led(2,15,0x00202000);     // yellow
  set_led(3,15,0x00200000);     // green
  set_led(4,15,0x0020000c);     // blue-green
  set_led(5,15,0x00100020);     // cyan
  set_led(6,15,0x00000020);     // blue
  set_led(7,15,0x00001020);     // violet

  b = 0x08;
  for(int i = 0; i < 16; i++)
    set_led(i,i,65536 * b);    // green

  for(int i = 0; i < 16; i++)
    set_led(15-i,16+i,256 * b);   // red

  for(int i = 0; i < 16; i++)
    set_led(16+i,15-i,(65536 + 256) * b);  // yellow

  for(int i = 0; i < 16; i++)
    set_led(16+i,16+i,257 * b);  // magenta

  write_leds();
  delay(500);
}
```

10.12 Configurable Math Channels

This is a configurable math channel for 6 functions with 4 control signals:

Required ICs:
2x DG409 4:1 analog switch
1x DG213 analog SPDT switch
1x AD633 multiplier
1x TL082

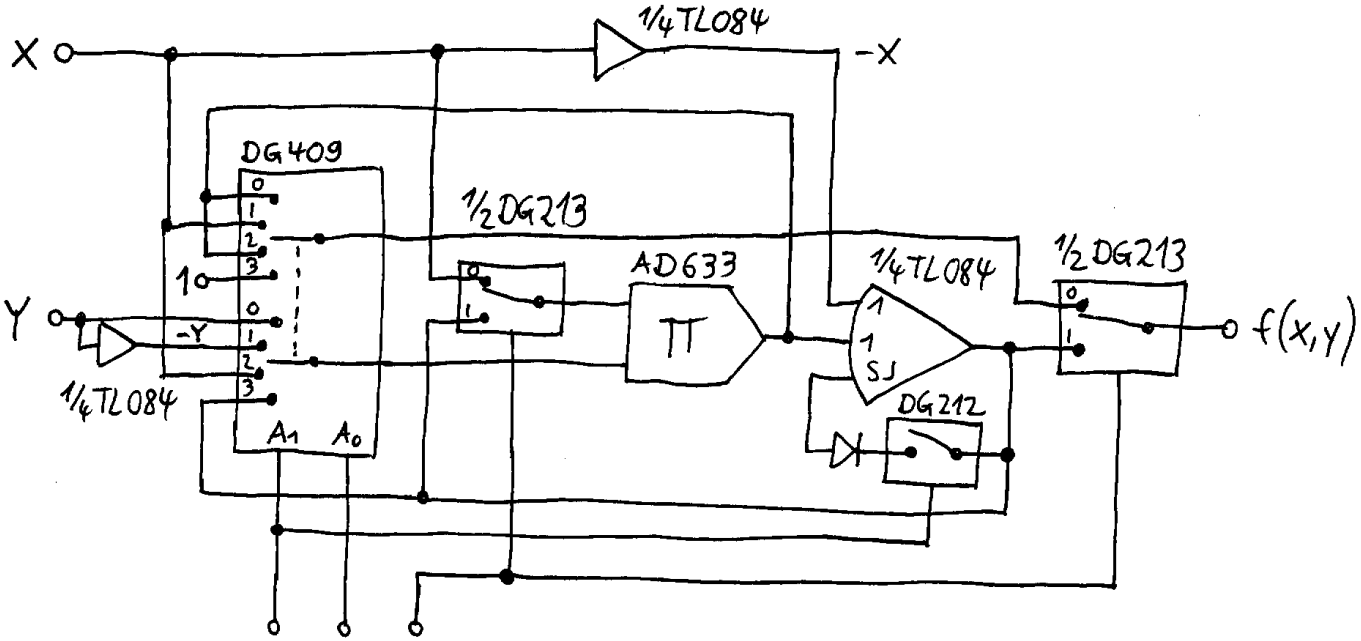


f(x,y)=				
X	0	0	0	0
x^2	0	0	0	1
XY	0	1	0	1
$\sqrt{x}$	1	0	1	0
X/Y	0	1	1	0
-X	0	0	1	1

This is a configurable math channel for 7 functions with 3 control signals:

Required ICs:
1x DG409 dual 4:1 analog switch
1x DG213 analog SPDT switch
1x DG212 analog SPST switch
1x AD633 multiplier
1x TL084

The 8th function (x/x) isn't useful.



0	0	0	XY
0	0	1	X/Y (Y>0)
0	1	0	X
0	1	1	-X/Y (Y<0)
1	0	0	X^2
1	0	1	–
1	1	0	1
1	1	1	$\sqrt{X}$

This is a configurable math channel for 8 functions with 3 control signals:

Required ICs:
1x DG409 dual 4:1 analog switch
2x DG213 analog SPDT switch
1x AD633 multiplier
1x TL082

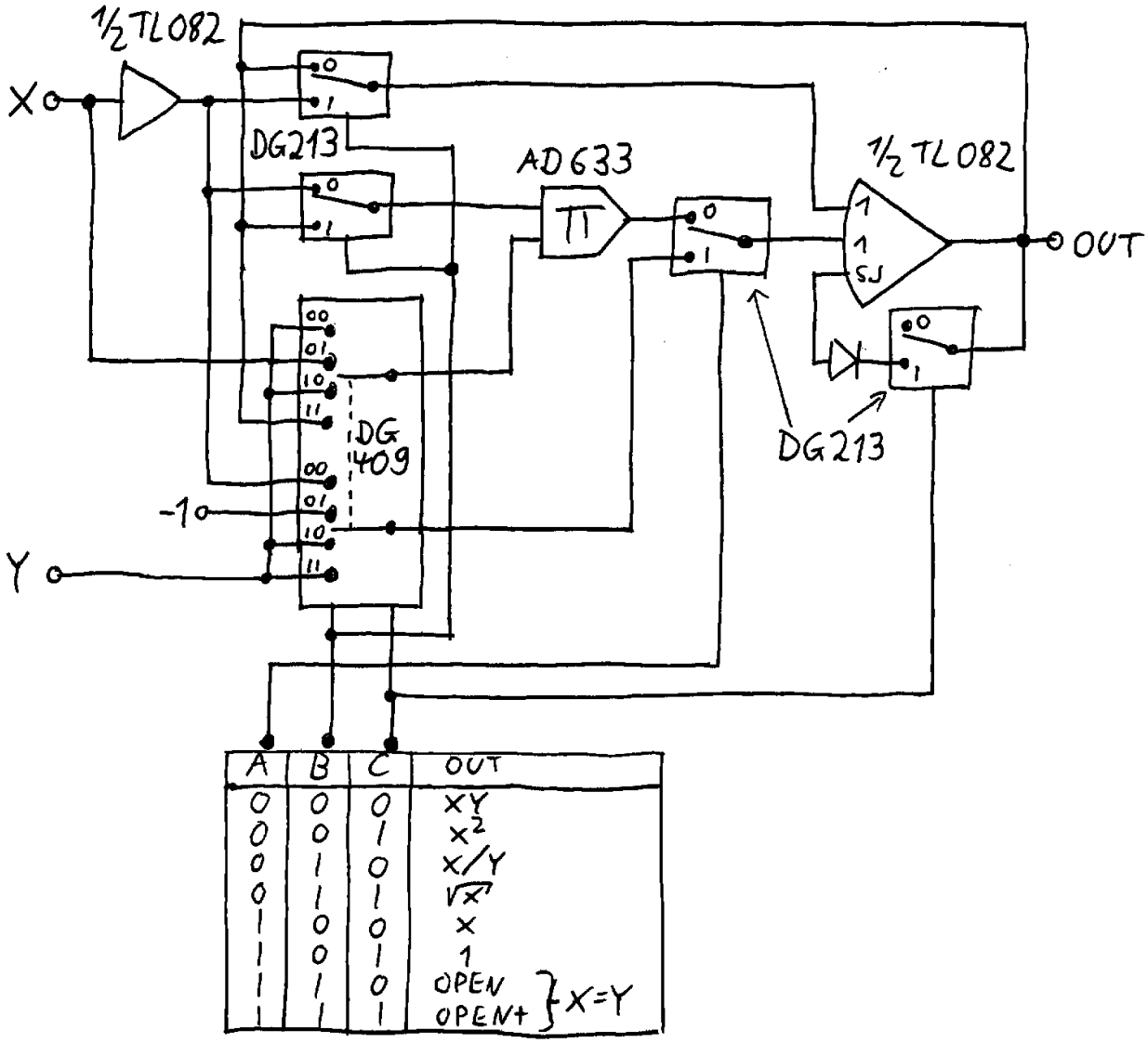
The last two functions (OPEN and OPEN+) are open amplifiers for X = Y, where X is the non-inverting and Y is the inverting input.

"OPEN+" means it has a feedback diode which forces the output to non-negative values.

Additional 3 control bits can be used for selecting 1 of 7 feedback capacitors with a DG408 (8:1 analog switch), to prevent oscillations of the open amplifier.

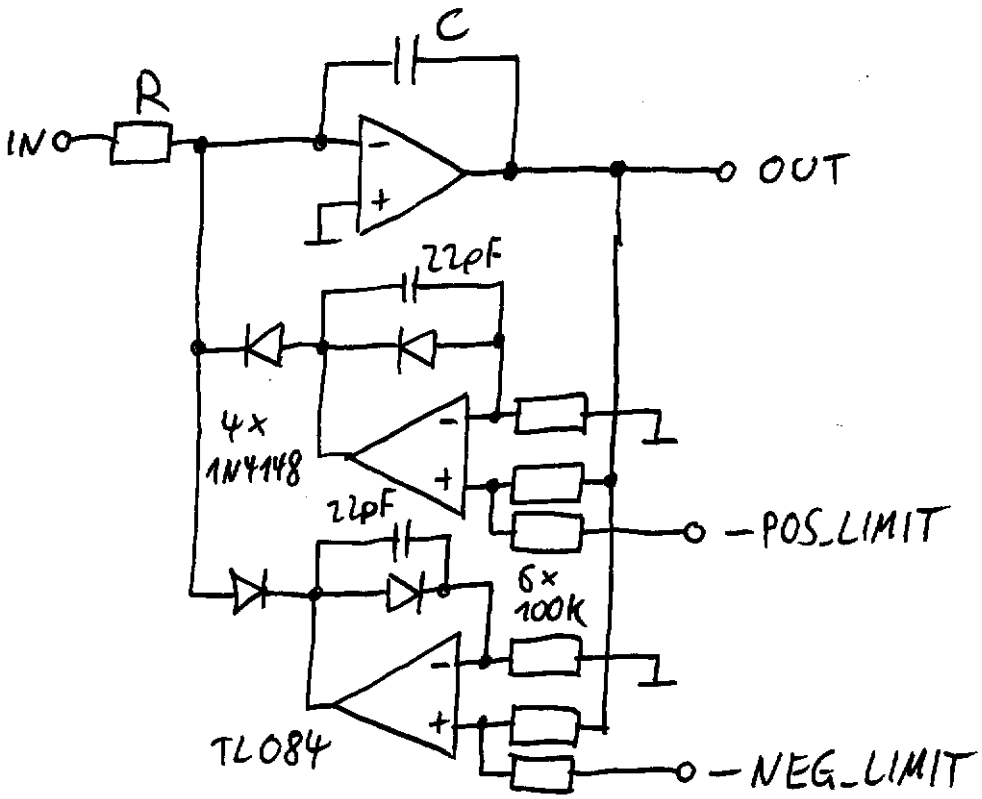
Capacitors and time constants (for R=100kΩ)

none	**--**
20pF	**2 µs**
50pF	**5 µs**
100pF	**10 µs**
200pF	**20 µs**
500pF	**50 µs**
1nF	**100 µs**
2nF	**200 µs**



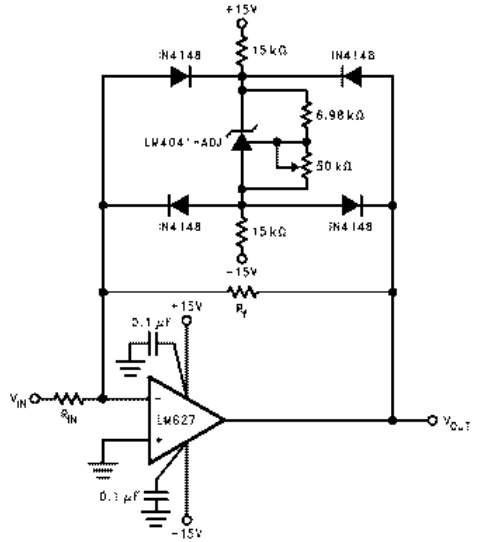
A	B	C	OUT
0	0	0	XY
0	0	1	X^2
0	1	0	X/Y
0	1	1	$\sqrt{X}$
1	0	0	X
1	0	1	1
1	1	0	OPEN } X=Y
1	1	1	OPEN+ } X=Y

Integrator with negative and positive limiters:



I found this limiter circuit in the LM4041 datasheet:
https://www.ti.com/lit/ds/symlink/lm4041-n.pdf

9.2.3 Bounded Amplifier



Bounded amplifier reduces saturation-induced delays and can prevent succeeding stage damage. Nominal clamping voltage is $\pm V_O$ (the reverse breakdown voltage of the LM4041-N) +2 diode V_F.

Figure 20. Bounded Amplifier

9.2.3.1 Design Requirements

Design an amplifier with output clamped at ±11.5 V.

9.2.3.2 Detail Design Procedure

With amplifier rails of ±15 V, the output can be bound to ±11.5 V with the LM4041 adjustable set for 10 V and two nominal diode voltage drops of 0.7 V.

$$V_{OUTBOUND} = 2 \times V_{FWD} + V_Z \quad (9)$$

$$V_{OUTBOUND} = 1.4\ V + 10\ V \quad (10)$$

Select R_S = 15 kΩ to keep L_R low. Calculate L_R to confirm R_S selection.

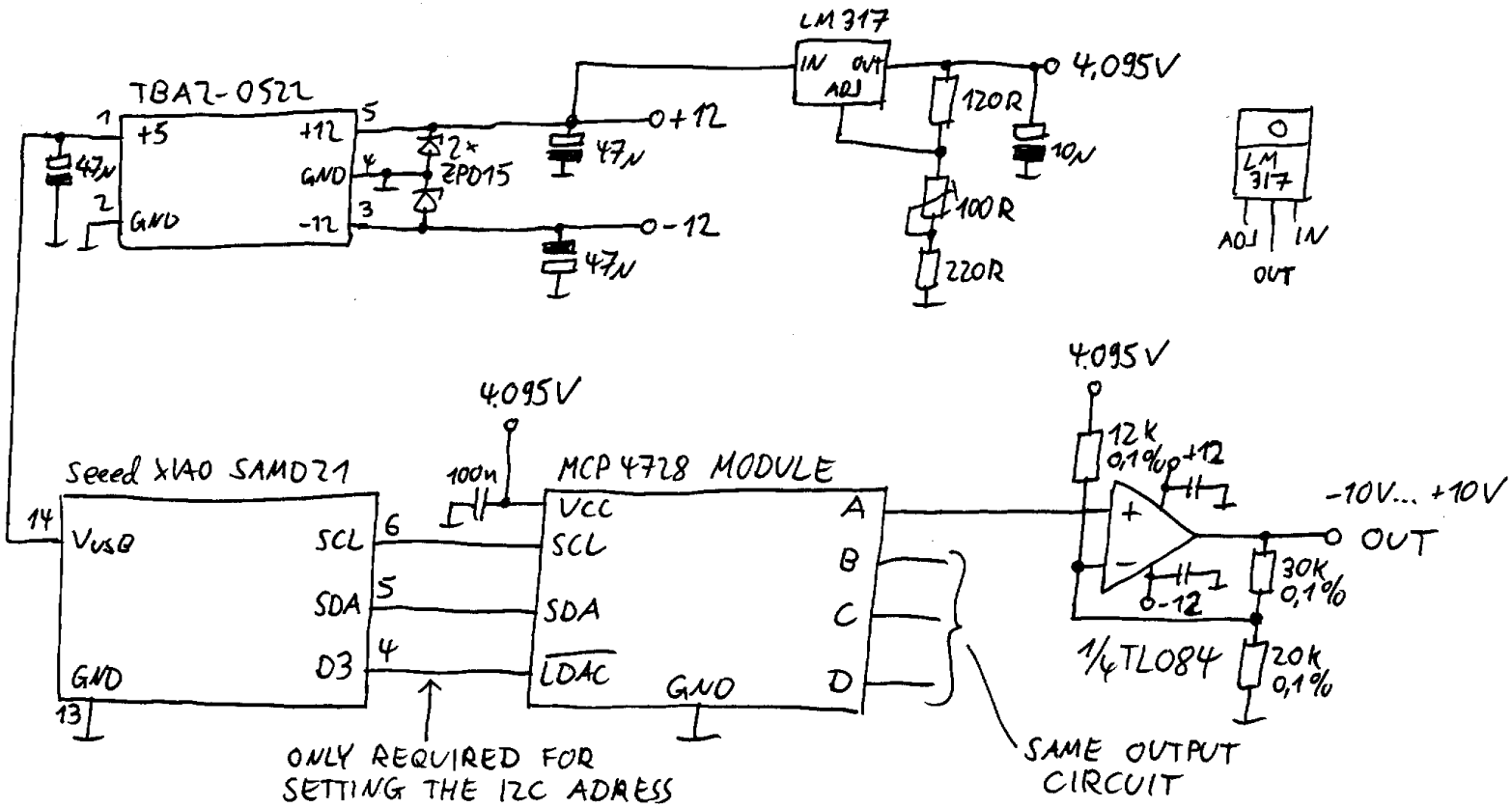
Use Equation 11, but in this case, take the negative supply into account.

$$I_R = (V_{IN} - V_{OUT}) / R \quad (11)$$

$$I_R = (V_{IN+} - V_{IN} - V_{OUT}) / R = (30\ V - 10\ V) / (R_{S1} + R_{S2}) = 20\ V / 30\ k\Omega = 0.667\ mA \quad (12)$$

This is an acceptable value for I_R that does not draw excessive current, but prevents the part from being starved for current.

10.13 Generator for -10 V to +10 V Constants



In this circuit the MCP4728 module gets a precision 4.095V supply voltage, which is also used as reference.

Codes from 48 to 2048 to 4048 are used for -10 V to 0 V to +10 V output. The output voltages are within +-0.5% of the expected values.

This is a non-inverting unipolar to bipolar converter.

Software for testing, with Adafruit_MCP4728 library:

```
#include <Adafruit_MCP4728.h>

Adafruit_MCP4728 mcp;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10);

  Serial.println("Adafruit MCP4728 test!");

  if (!mcp.begin())
  {
    Serial.println("Failed to find MCP4728 chip");
    while (1);
  }
}

void loop()
{
  mcp.setChannelValue(MCP4728_CHANNEL_A, 48);     // -10 V
  mcp.setChannelValue(MCP4728_CHANNEL_B, 2048);   //   0 V
  mcp.setChannelValue(MCP4728_CHANNEL_C, 3048);   //  +5 V
  mcp.setChannelValue(MCP4728_CHANNEL_D, 4048);   // +10 V
  delay(100);
}
```

Software for testing, without Adafruit_MCP4728 library:

```
#include <Wire.h>

int addr;
int x;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10);

  Wire.begin();
```

```
}

void loop()
{
  addr = 0x60;
  x = 48;                          // -10 V
  Wire.beginTransmission(addr);
  Wire.write(lowByte(0x40));
  Wire.write(highByte(x));
  Wire.write(lowByte(x));
  Wire.endTransmission();

  x = 2048;                        //   0 V
  Wire.beginTransmission(addr);
  Wire.write(lowByte(0x42));
  Wire.write(highByte(x));
  Wire.write(lowByte(x));
  Wire.endTransmission();

  x = 3048;                        //  +5 V
  Wire.beginTransmission(addr);
  Wire.write(lowByte(0x44));
  Wire.write(highByte(x));
  Wire.write(lowByte(x));
  Wire.endTransmission();

  x = 4048;                        // +10 V
  Wire.beginTransmission(addr);
  Wire.write(lowByte(0x46));
  Wire.write(highByte(x));
  Wire.write(lowByte(x));
  Wire.endTransmission();
  delay(100);
}
```

Non-inverting unipolar to bipolar converter for MCP4728 rail-to-rail DAC:

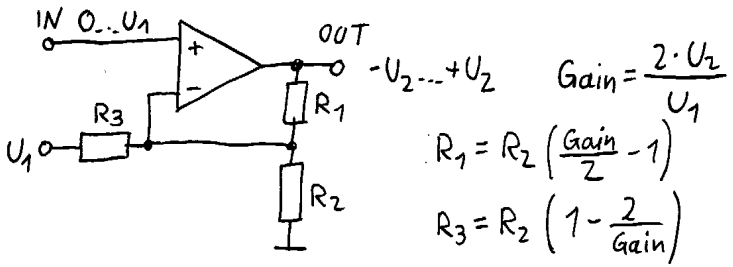
Example 3:

Gain = 3, U1 = 4.095 V

R1	R2	R3
1	0.6666	0.4
1.5	1	0.6
2	1.3333	0.8
2.5	1.6666	1
3	2	1.2
5	3.3333	2
7.5	5	3
10	6.6666	4
15	10	6

See also: https://www.ti.com/lit/ug/slau525/slau525.pdf?ts=1729510873645

See also (for unity gain):
https://www.analog.com/en/resources/technical-articles/challenge-the-conventional--make-unipolar-dacs-bipolar.html



Example 1:

$U_1 = 3.333\,V$ $U_2 = 10\,V$ $Gain = 6$

$R_1 = 6\,k\Omega$ $R_2 = 3\,k\Omega$ $R_3 = 2\,k\Omega$

Example 2:

At the input is a 12-bit rail-to-rail DAC.

$U_1 \mathrel{\hat{=}} 4095$ steps

But only 4000 steps of the DAC shall be used:

48... 2048... 4048

for output: -10V... 0V... +10V in 5mV steps

If the gain is 6 (as in example 1), then the supply voltage of the DAC can be calculated:

$$U = \frac{2 \cdot U_2}{Gain} \cdot \frac{4095}{4000} = 3.4125\,V$$

This is the inverting unipolar to bipolar converter:

U_{IN}	U_{OUT}
0 V	10.24 V
2.048 V	0 V
4.096 V	-10.24 V

Some suitable resistor combinations for 0 to U_1 = 4.096V input and -10.24 to 10.24V output (Gain = 5):

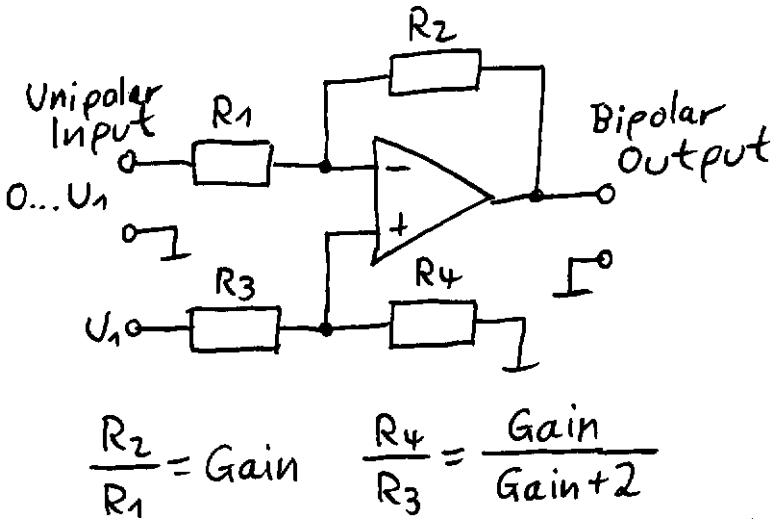
R1	R2	R3	R4
100 kΩ	500 kΩ	140 kΩ = 120 kΩ + 20 kΩ	100 kΩ
200 kΩ	1 MΩ	168 kΩ = 100 kΩ + 68 kΩ	120 kΩ
400 kΩ	2 MΩ	560 kΩ	400 kΩ

This is the inverting bipolar to unipolar converter:

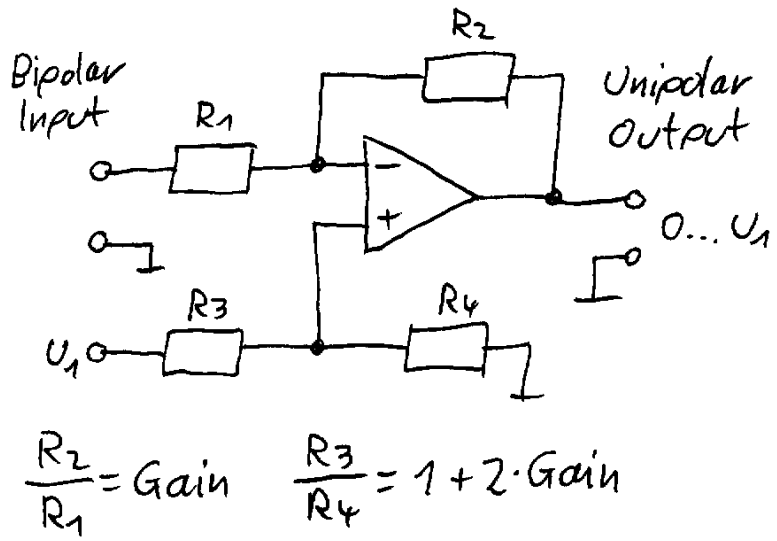
U_{IN}	U_{OUT}
10.24 V	0 V
0 V	2.048 V
-10.24 V	4.096 V

Some suitable resistor combinations for -10.24 to 10.24V input and 0 to U_1 = 4.096V output (Gain = 0.2):

R1	R2	R3	R4
500 kΩ	100 kΩ	140 kΩ = 120 kΩ + 20 kΩ	100 kΩ
1 MΩ	200 kΩ	168 kΩ = 100 kΩ + 68 kΩ	120 kΩ
2 MΩ	400 kΩ	560 kΩ	400 kΩ



$$\frac{R_2}{R_1} = Gain \qquad \frac{R_4}{R_3} = \frac{Gain}{Gain+2}$$



$$\frac{R_2}{R_1} = Gain \qquad \frac{R_3}{R_4} = 1 + 2 \cdot Gain$$

10.14 Change the I2C Address of the MCP4728 Chip

https://github.com/jknipper/mcp4728_program_address/blob/master/mcp4728_program_address.ino (I didn't get this working)

Doesn't run with SAMD21 or with Teensy 4.0.

Might work with this library: https://github.com/felias-fogg/SlowSoftI2CMaster (I didn't get this working, many error messages)

But this program works fine, found here:
https://forum.pjrc.com/index.php?threads/programming-the-i2c-address-of-a-mcp4728-4-channel-dac-using-i2c_t3.28424/

```
/*
  Sets the MCP4728 DAC address to a new value by using the LDAC pin
  Adjust the pins you will use for the SDA and SCL bus
  Adjust the IO line you will use for the LDAC signal
  Just run this once on the DAC you want to remap the address of.
  Or include in a larger project and call as required...
 */

uint8_t SCLpin = D5;
uint8_t SDApin = D4;
uint8_t LDACpin = D3;                   // pulled LOW on PCB so drive when needed Hi
uint8_t LEDpin = 13;                    // Teensy LED (Hi logic = lit)

uint8_t I2C_OldAddress = 0;             // normally 0
uint8_t I2C_NewAddress = 1;             // any value 0-7

int qflash = 50;          // 50ms
int sflash = 250;         // 250ms
int LEDhold = 1000;       // 1s
int I2Cdelay = 1;         // 1ms
int FailByte = 0;         // which byte we failed at (if we failed)
int Success;

void setup() {

      uint8_t flish;
      uint8_t outchar;
      uint8_t tmpchar;
```

```
    Serial.begin(115200);     // initialize serial interface for print()

    Success = 0;        // until it works

    // initialize the digital pins as an output
    pinMode(SCLpin, OUTPUT);
    pinMode(SDApin, OUTPUT);
    pinMode(LDACpin, OUTPUT);
    pinMode(LEDpin, OUTPUT);

    digitalWrite(SCLpin, HIGH);      // SCL + SDA high to start
    digitalWrite(SDApin, HIGH);      // SCL + SDA high to start
    digitalWrite(LDACpin, LOW);      // LDAC Low to start

    // Inform the operation to be done...
    Serial.println("Setting DAC Address as follows:");
    Serial.print("Existing DAC Address = ");
    Serial.println(I2C_OldAddress, DEC);
    Serial.print("New DAC Address = ");
    Serial.println(I2C_NewAddress, DEC);
  Serial.println("");

    // Flash LED number of times for new address (double flash then 0 to 7 flashes)
    for(flish = 0; flish<2; flish++){
          digitalWrite(LEDpin, HIGH);
          delay(qflash);
          digitalWrite(LEDpin, LOW);
          delay(qflash);
    }
    delay(LEDhold);
    for(flish = 0; flish<I2C_NewAddress; flish++){
          digitalWrite(LEDpin, HIGH);
          delay(sflash);
          digitalWrite(LEDpin, LOW);
          delay(sflash);
    }
    delay(LEDhold);

    // set LDAC Hi
    digitalWrite(LDACpin, HIGH);     // LDAC Low to start
    delay(I2Cdelay);
    // send a start condition
    digitalWrite(SDApin, LOW);
```

```
	delay(I2Cdelay);
	digitalWrite(SCLpin, LOW);
	delay(I2Cdelay);
	// byte 1
	outchar = I2C_OldAddress;
	outchar <<= 1;                  // shift L by 1
	outchar |= 0xc0;                // or in 0x60 to address bits
	tmpchar = I2CwriteByte(outchar);       // write byte 1 (expect slave to ACK and returns 1 if it does, 0 of not)
	FailByte = 1;
	if (tmpchar == 1) {       // success
		// byte 2
		outchar = I2C_OldAddress;
		outchar <<= 2;                  // shift L by 2
		outchar |= 0x61;                // or in 0x61 to set the address change command
		tmpchar = LDACwriteByte(outchar);      // funky write byte 2 which sets LDAC low after the 8th clock
		                                //  (expect slave to ACK and returns 1 if it does, 0 of not)
		FailByte = 2;
		if (tmpchar == 1) {       // success
			// byte 3
			outchar = I2C_NewAddress;
			outchar <<= 2;            // shift L by 2
			outchar |= 0x62;          // or in 0x62 to set the address change command and signify byte 3 (b1 set, b0 clear)
			tmpchar = I2CwriteByte(outchar);       // write byte 3 (expect slave to ACK only if written correctly)
			FailByte = 3;
			if (tmpchar == 1) {       // success
				// byte 4
				outchar = I2C_NewAddress;
				outchar <<= 2;                          // shift L by 2
				outchar |= 0x63;    // or in 0x63 to set the address change command and signify byte 4 (b1 set, b1 set)
				tmpchar = I2CwriteByte(outchar);       // write byte 4 (expect slave to ACK)
				FailByte = 4;
				if (tmpchar == 1) {       // success
					Success = 1;
					FailByte = 0;
				}
			}
		}
	}
	// send a stop condition
	// SCL will be Low
	digitalWrite(SDApin, LOW);// SDA low in case it's Hi
	delay(I2Cdelay);
	digitalWrite(SCLpin, HIGH);
```

```
	delay(I2Cdelay);
	digitalWrite(SDApin, HIGH);	// SCL Hi
	delay(I2Cdelay);

	// now set the LDAC low in case it is still high (failed on byte 1)
	digitalWrite(LDACpin, LOW);	// LDAC Low
	// now announce the status...
	if (Success == 1) {
		Serial.println("DAC Address succesfully set to new value.");
	}
	else {
		Serial.print("DAC Address Set procedure FAILED!!  I2C ACK was missing at Byte number ");
		Serial.println(FailByte, DEC);
	}
}

uint8_t I2CwriteByte(uint8_t outchar){
	// Enters with SCL LOW, SDA undefined Hi or Low
	// SDA set to output
	//
	// Leaves with SCL LOW, SDA HIGH
	// SDA set to output
	//
	// returns 1 if acked OK, 0 if not

	int shuffle;
	uint8_t MyAck;
	int tempint;

	shuffle = 9;		// 8 data bits
	while (--shuffle) {
		if (outchar & 0x80)
			digitalWrite(SDApin, HIGH);
		else
			digitalWrite(SDApin, LOW);
		delay(I2Cdelay);
		digitalWrite(SCLpin, HIGH);
		delay(I2Cdelay);
		outchar <<= 1;			// shift up data for bext bit
		digitalWrite(SCLpin, LOW);
		delay(I2Cdelay);
	}
	// Completed sending byte, now allow slave to ACK
```

```
	pinMode(SDApin, INPUT);                 // turn SDA to input for ACK
	digitalWrite(SCLpin, HIGH);
	delay(I2Cdelay);
	// ACK received here with a bit of luck (we may test some day)
	// read input line
	tempint = digitalRead(SDApin);
	if (tempint == 0) {        // acked OK
		MyAck = 1;
	}
	else {        // failed
		MyAck = 0;
	}
	digitalWrite(SCLpin, LOW);
	delay(I2Cdelay);
	digitalWrite(SDApin, HIGH);             // set output register high before shanging DDR
	pinMode(SDApin, OUTPUT);                // turn SDA back to an output
	digitalWrite(SDApin, HIGH);             // set output high to exit
	delay(I2Cdelay);
	return MyAck;
}

uint8_t LDACwriteByte(uint8_t outchar){
	// Enters with SCL LOW, SDA undefined Hi or Low, LDAC High
	// SDA set to output
	//
	// Leaves with SCL LOW, SDA HIGH, LDAC Low
	// SDA set to output
	//
	// returns 1 if acked OK, 0 if not

	int shuffle;
	uint8_t MyAck;
	int tempint;

	shuffle = 9;        // 8 clocks
	while (--shuffle) {
		if (outchar & 0x80)
			digitalWrite(SDApin, HIGH);
		else
			digitalWrite(SDApin, LOW);
		delay(I2Cdelay);
		digitalWrite(SCLpin, HIGH);
		delay(I2Cdelay);
```

```
            outchar <<= 1;                                          // shift up data for bext bit
            digitalWrite(SCLpin, LOW);
            delay(I2Cdelay);
      }
      // Now we drop the LDAC pin to initiate the EEPROM command
      digitalWrite(LDACpin, LOW);      // LDAC Low
      delay(I2Cdelay);
      // Completed sending byte, now allow slave to ACK
      pinMode(SDApin, INPUT);                 // turn SDA to input for ACK
      digitalWrite(SCLpin, HIGH);
      // ACK received here with a bit of luck (we may test some day)
      // read input line
      tempint = digitalRead(SDApin);
      if (tempint == 0) {       // acked OK
            MyAck = 1;
      }
      else {       // failed
            MyAck = 0;
      }
      delay(I2Cdelay);
      digitalWrite(SCLpin, LOW);
      delay(I2Cdelay);
      digitalWrite(SDApin, HIGH);             // set output register high before shanging DDR
      pinMode(SDApin, OUTPUT);                // turn SDA back to an output
      digitalWrite(SDApin, HIGH);             // set output high to exit
      delay(I2Cdelay);
      return MyAck;
}

void loop() {
      // continually announce the status...
      if (Success == 1) {
            Serial.println("DAC Address succesfully set to new value.");
      }
      else {
            Serial.print("DAC Address Set procedure FAILED!!  I2C ACK was missing at Byte number ");
            Serial.println(FailByte, DEC);
      }
      delay(3000);               // wait for a few seconds
      // Our work is done.  Power cycle and programme another
}
```

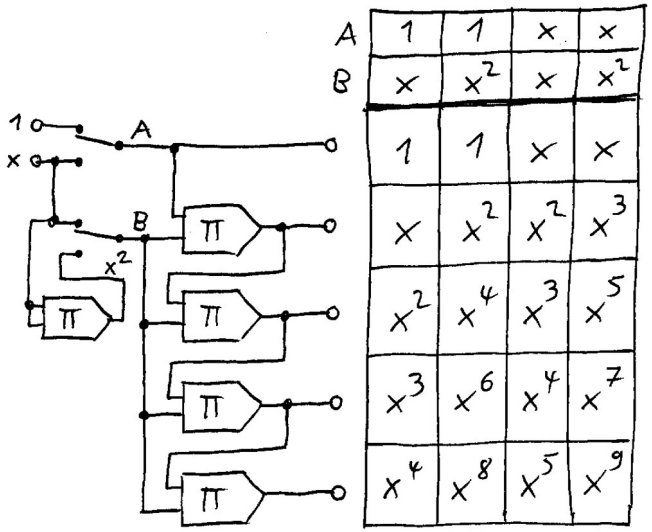
10.15 Ideas for Integrators

- **Multiple time constants**
- **Adjustable positive and negative limiters**
- **Multiply before integrate (for example for sine generators, or for continuous forward/backward time scaling)**
- **Initial conditions**
- **Start - stop**

10.16 Ideas for Comparators

- **abs(x)**
- **positive output only (out = x, if x > 0 out = 0, if x < 0)**
- **negative output only (out = x, if x < 0 out = 0, if x > 0)**
- **min(x,y) = positive limiter**
- **max(x,y) = negative limiter**
- **sgn(x) (output a or b, a and b are adjustable constants)**
- **sgn(x+y) (output a or b, a and b are adjustable constants)**
- **sgn(x) same as above, but with feedback, output is summed to input**
- **sgn(x) (output 0 or a, a is an adjustable constant)**
- **sgn(x+y) (output 0 or a, a is an adjustable constant)**
- **out = x, if z > 0, out = y, if z < 0**
- **limiter with adjustable limits a and b**

10.17 Math Block for Power Series Approximation



A \ B	1	x	x^2	x^3
1	1	x	x^2	x^3
x	x	x^2	x^3	x^4
x^2	x^2	x^3	x^4	x^5
x^3	x^3	x^4	x^5	x^6
x^4	x^4	x^5	x^6	x^7

	1	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9
sin(x)		1		-0.1667		0.0083		-0.0002		
cos(x)	1		-0.5		0.0417		-0.0014			
tan(x) for |x| < 1.5707		1		0.3333		0.1333		0.0540		0.0219
arcsin(x) for |x| < 1		1		0.1667		0.075		0.0446		
arccos(x) for |x| < 1	1.5708	-1		-0.1667		-0.075		-0.0446		
arctan(x) for |x| < 1		1		-0.3333		0.2		-0.1429		
exp(x)	1	1	0.5	0.1667	0.0417	0.0083	0.0014	0.0002		
ln(x)										

10.18 Math Block for sin / cos / tan / arctan

10.19 LCD Display 40x4 Characters with I2C Interface

The display is available here: https://de.aliexpress.com/item/32908727195.html
The library is available from Surenoo: https://www.surenoo.com/

Demo program:

```
#include "LiquidCrystal_I2C_4004.h"      // file in "" is searched in local folder

const byte lcdAddr = 0x27;  // Address of I2C backpack
const byte lcdCols = 40;    // Number of character in a row
const byte lcdRows = 4;     // Number of lines
LiquidCrystal_I2C lcd(lcdAddr, lcdCols, lcdRows); // set the LCD address to 0x27 for a 16 chars and 2 line display

void setup()
{
  lcd.begin();
  delay(1000);
  lcd.home();
  lcd.backlight();
  lcd.setCursor(0, 0); lcd.print("...cursor.....");
  lcd.cursor(); delay(2000);
  lcd.setCursor(0, 0); lcd.print("...noCursor...");
  lcd.noCursor(); delay(2000);
  lcd.setCursor(0, 0); lcd.print(".backlight off..");
  lcd.noBacklight(); delay(2000);
  lcd.setCursor(0, 0); lcd.print(".backlight on..");
  lcd.backlight(); delay(2000);
  lcd.setCursor(0, 0); lcd.print("Test 0");
  lcd.setCursor(1, 1); lcd.print("Test 1");
  lcd.setCursor(2, 2); lcd.print("Test 2");
  lcd.setCursor(3, 3); lcd.print("Test 3");
  delay(1000);
  lcd.clear(0);   // clears lines 0 and 1
  delay(1000);
  lcd.clear(1);   // clears lines 2 and 3
}

void loop() { }
```

10.20 SI Units

Exponents					Unit	Physical Quantity	Notes
s	m	kg	A	K			
					rad	Radiant	
1					s	Time	
	1				m	Length	
		1			kg	Mass	
			1		A	Current	
				1	K	Temperature	
-1					Hz	Frequency	
-2	1	1			N	Force	
-1	1				$m\ s^{-1}$	Velocity	
-2	1				$m\ s^{-2}$	Accelertion	
-3	2	1			$W = J\ s^{-1}$	Power	
-2	2	1			J = W s = N m	Energy, Work	Problem: Energy, Work and Torque have the same unit
-2	2	1			N m	Torque	
-2	-1	1			Pa	Pressure	
1			1		C	Electric Charge	
-3	2	1	-1		V	Voltage	
-3	2	1	-2		Ω	Resistance	
4	-2	-1	2		F	Capacitance	
-2	2	1	-2		H	Inductance	
-2	2	1	-1		Wb	Magnetic Flux	
-2		1	-1		T	Magnetic Flux Density	

11 Analog Compiler

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Units_Test
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            calc();
        }

        private void numericUpDown2_ValueChanged(object sender, EventArgs e)
        {
            calc();
        }

        private void numericUpDown3_ValueChanged(object sender, EventArgs e)
        {
            calc();
        }

        private void numericUpDown4_ValueChanged(object sender, EventArgs e)
        {
            calc();
        }
```

```
        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            calc();
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            calc();
        }

        void calc()
        {
            int s = (int)numericUpDown1.Value;
            int m = (int)numericUpDown2.Value;
            int kg = (int)numericUpDown3.Value;
            int A = (int)numericUpDown4.Value;

            richTextBox1.Clear();

            Signal sig = new Signal("test", 20, s, m, kg, A, false, radioButton2.Checked);
            richTextBox1.SelectedRtf = sig.getScalingAsRTF();
            richTextBox1.AppendText("\n");
            richTextBox1.SelectedText = sig.getScalingAsString();
        }

    }
}
```

Signal.cs

```
using System;

namespace Units_Test
{
    class Signal
    {
        public string name;
        public float scaling;
        public int s;
```

```
        public int m;
        public int kg;
        public int A;
        public Boolean scalingIsGiven;
        public Boolean torque;
        public System.Collections.Generic.List<PhysicalUnit> physicalUnits;

        class UnitExponent : IComparable<UnitExponent>
        {
            public string name { get; set; }
            public int e { get; set; }

            public UnitExponent(string name, int e)   // Konstruktor
            {
                this.name = name;
                this.e = e;
            }

            public int CompareTo(UnitExponent comparePart)   // Default comparer for Part type.
            {
                // A null value means that this object is greater.
                if (comparePart == null)
                    return 1;
                else
                    return this.e.CompareTo(comparePart.e);
            }
        }

        public class PhysicalUnit
        {
            public string name;
            public int s;
            public int m;
            public int kg;
            public int A;
            public Boolean baseSI;

            public PhysicalUnit(string name, int s, int m, int kg, int A, Boolean baseSI)   // Konstruktor
            {
```

```
            this.name = name;
            this.s = s;
            this.m = m;
            this.kg = kg;
            this.A = A;
            this.baseSI = baseSI;
        }
    }

    public Signal(string name, float scaling, int s, int m, int kg, int A, Boolean scalingIsGiven, Boolean torque)   // Konstruktor
    {
        this.name = name;
        this.scaling = scaling;
        this.s = s;
        this.m = m;
        this.kg = kg;
        this.A = A;
        this.scalingIsGiven = scalingIsGiven;
        this.torque = torque;

        physicalUnits = new System.Collections.Generic.List<PhysicalUnit>();
        physicalUnits.Add(new PhysicalUnit("s",   1,  0,  0,  0, true));
        physicalUnits.Add(new PhysicalUnit("m",   0,  1,  0,  0, true));
        physicalUnits.Add(new PhysicalUnit("kg",  0,  0,  1,  0, true));
        physicalUnits.Add(new PhysicalUnit("A",   0,  0,  0,  1, true));
        physicalUnits.Add(new PhysicalUnit("Hz", -1,  0,  0,  0, false));
        physicalUnits.Add(new PhysicalUnit("N",  -2,  1,  1,  0, false));
        physicalUnits.Add(new PhysicalUnit("W",  -3,  2,  1,  0, false));
        physicalUnits.Add(new PhysicalUnit("J",  -2,  2,  1,  0, false));
        physicalUnits.Add(new PhysicalUnit("Pa", -2, -1,  1,  0, false));
        physicalUnits.Add(new PhysicalUnit("C",   1,  0,  0,  1, false));
        physicalUnits.Add(new PhysicalUnit("V",  -3,  2,  1, -1, false));
        physicalUnits.Add(new PhysicalUnit("Ω",  -3,  2,  1, -2, false));
        physicalUnits.Add(new PhysicalUnit("S",   3, -2, -1,  2, false));
        physicalUnits.Add(new PhysicalUnit("F",   4, -2, -1,  2, false));
        physicalUnits.Add(new PhysicalUnit("H",  -2,  2,  1, -2, false));
        physicalUnits.Add(new PhysicalUnit("Wb", -2,  2,  1, -1, false));
        physicalUnits.Add(new PhysicalUnit("T",  -2,  0,  1, -1, false));
    }
```

```
        public string getScalingAsRTF()
        {
            string unit = "";
            Boolean first;

            foreach (PhysicalUnit pu in physicalUnits)      // check if the unit equals a physical unit
            {
                if ((s == pu.s) && (m == pu.m) && (kg == pu.kg) && (A == pu.A))
                {
                    unit += pu.name;
                    break;      // one match is enough!
                }
            }
            if ((unit == "J") && (torque == true))
                unit = @"N\u8901?m";    // replace "J" by "N*m"  (with multiplication dot)
            if (unit == "Ω")
                unit = @"\u0937?";

            if (unit == "")   // if it's not a physical unit, then try to express the unit as a physical unit with additional "s^exponent"
            {
                foreach (PhysicalUnit pu in physicalUnits)
                {
                    if ((m == pu.m) && (kg == pu.kg) && (A == pu.A))
                    {
                        if ((pu.baseSI == false) && (pu.name != "Hz"))   // exclude the base SI units and Hz
                        {
                            {
                                if (pu.name == "Ω")
                                    unit = @"\u0937?\u8901?s";
                                else
                                    unit += pu.name + @"\u8901?s";
                                if (s - pu.s != 1)
                                    unit += @"\super " + (s - pu.s).ToString() + @"\nosupersub ";
                                break;      // one match is enough!
                            }
                        }
                    }
                }
```

```
        }

        if (unit == "")   // if all of the above was not successful, then express the unit in base SI units
        {
            System.Collections.Generic.List<UnitExponent> exponents = new System.Collections.Generic.List<UnitExponent>();
            exponents.Add(new UnitExponent("s", s));
            exponents.Add(new UnitExponent("m", m));
            exponents.Add(new UnitExponent("kg", kg));
            exponents.Add(new UnitExponent("A", A));
            exponents.Sort();     // sort by size of exponent
            exponents.Reverse();  // largest exponents first

            first = true;
            foreach (UnitExponent ex in exponents)     // express the unit in base SI units, sorted by exponents
            {
                if (ex.e != 0)
                {
                    if(first == false)
                        unit += @"\u8901?";    // multiplication dot
                    unit += ex.name;
                    if (ex.e != 1)
                        unit += @"\super " + ex.e.ToString() + @"\nosupersub ";
                    first = false;
                }
            }
        }

        return (@"{\rtf1\ansi " + scaling.ToString() + " " + unit + "}");
    }

    public string getScalingAsString()
    {
        string unit = "";
        foreach (PhysicalUnit pu in physicalUnits)      // check if the unit equals a physical unit
        {
            if ((s == pu.s) && (m == pu.m) && (kg == pu.kg) && (A == pu.A))
            {
                unit += pu.name;
                break;      // one match is enough!
```

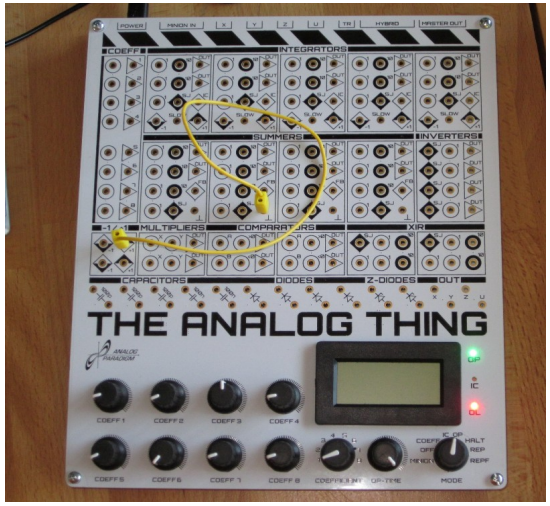
```
            }
        }
        if ((unit == "J") && (torque == true))
            unit = "Nm";    // replace "J" by "Nm"

        if (unit == "")   // if it's not a physical unit, then try to express the unit as a physical unit with additional "s^exponent"
        {
            foreach (PhysicalUnit pu in physicalUnits)
            {
                if ((m == pu.m) && (kg == pu.kg) && (A == pu.A))
                {
                    if ((pu.baseSI == false) && (pu.name != "Hz"))   // exclude the base SI units and Hz
                    {
                        {
                            unit += pu.name + " s";
                            if (s - pu.s != 1)
                                unit += "^" + (s - pu.s).ToString();
                            break;      // one match is enough!
                        }
                    }
                }
            }
        }

        if (unit == "")   // if all of the above was not successful, then express the unit in base SI units
        {
            System.Collections.Generic.List<UnitExponent> exponents = new System.Collections.Generic.List<UnitExponent>();
            exponents.Add(new UnitExponent("s", s));
            exponents.Add(new UnitExponent("m", m));
            exponents.Add(new UnitExponent("kg", kg));
            exponents.Add(new UnitExponent("A", A));
            exponents.Sort();     // sort by size of exponent
            exponents.Reverse();  // largest exponents first

            foreach (UnitExponent ex in exponents)     // express the unit in base SI units, sorted by exponents
            {
                if (ex.e != 0)
                {
                    unit += ex.name;
```

```
                        if (ex.e != 1)
                            unit += "^" + ex.e.ToString();
                        unit += " ";
                    }
                }
            }

            return (scaling.ToString() + " " + unit);
        }
    }
}
```

12 Problems and Solutions

12.1 Display is off

If you ever have the problem that nothing is shown in the display, you should check if you have accidentally shorted the +1 signal to ground. The same thing may also happen if two or more outputs of summers or integrators are shorted.



12.2 Unexpected Overflows

I had some unexpected overflows when I tested a circuit. I did check with an oscilloscope that all outputs of the computing elememts were in the allowed range. But nevertheless after a few seconds in "OP" mode the overload LED was going on. Finally I figured out that the overload came from an unused integrator. Keep that in mind when you do a calculation in "OP" mode over a longer time. All unused integrators must be disabled by connecting their input to their output.

12.3 Integrators are drifting away too fast

If you ever have the problem that one if the integrators is drifting too fast into overload, it's time to clean the surface of the front panel with alcohol. In my case the overload came within 0.2 seconds. There must have been a fingerprint on the front panel. 200 MΩ · 1nF = 0.2s. After cleaning, it works again as expected.

12.4 THAT doesn't run after power-on

That's a known problem when THAT is in REPF mode during power-on. Switch to REP and back to REPF, then it will run.

12.5 Calibrate THATs Display

THATs display can be calibrated with the trimmer which is right of the MODE switch on the lower printed circuit board. Connect -1 or +1 to U and set the display to -1.000 or +1.000.

13 Unsolved Problems

- **Understand how the limiters in figures 5.20 and 5.21 in Bernd's book work.**
- **Understand GIT. No, I don’t really want to understand this complicated shit.**

14 Mathematics

14.1 Books about Mathematics

Calculus made easy (written in 1910): http://calculusmadeeasy.org/

Bronstein-Semendjajew: Taschenbuch der Mathematik, 21. Auflage, ISBN 3-87144-492-8

Feldmann, Dietrich: Repetitorium der Ingenieur-Mathematik, ISBN 3-923923-00-7

Feldmann, Dietrich: Repetitorium der Ingenieur-Mathematik, Teil 2, ISBN 3-923923-05-8

Mühlbach, Günter: Repetitorium der Ingenieur-Mathematik, Teil 3, ISBN 3-923923-30-9

14.2 Powers and Roots

$a^x = e^{x \ln(a)}$ $a^{-x} = 1 / a^x$ $(1 / a)^{-x} = a^x$ $a^0 = 1$ $(a / b)^{-n} = (b / a)^n$

$x^{a+b} = x^a \cdot x^b$ $x^{a-b} = x^a / x^b$ $x^{ab} = (x^a)^b = (x^b)^a$ $x^a = (x^{a/b})^b$ $x^{a/b} = \sqrt[b]{x^a}$ $x^{a/2} = \sqrt{x^a}$ $(a \cdot b)^x = a^x \cdot b^x$ $(a/b)^x = a^x / b^x$

$\sqrt{x} = x^{1/2}$ $\sqrt[n]{ab} = \sqrt[n]{a} \cdot \sqrt[n]{b}$ $\sqrt[n]{a/b} = \sqrt[n]{a} / \sqrt[n]{b}$ $\sqrt[n]{a} = a^{1/n}$ $\sqrt[n]{a^m} = (\sqrt[n]{a})^m = a^{m/n}$

$2 - \sqrt{3} = 1 / (2 + \sqrt{3})$

The expression 2^{3^x} (without parantheses) must be interpreted as $2^{(3^x)}$ and not as $(2^3)^x$. To clarify things, it's better to use parantheses.
See also this german Wikipedia page: https://de.wikipedia.org/wiki/Operatorrangfolge

14.3 Logarithms and Exponential Functions

$\log(a \cdot b) = \log(a) + \log(b)$ $\log(a / b) = \log(a) - \log(b)$ $\log(1 / x) = -\log(x)$

$\log(a^b) = b \cdot \log(a)$ $\log(x^2) = 2 \cdot \log(x)$

$\log(1) = 0$ $\ln(x) = \log_e(x)$

$\log_a(x) = \log_b(x) / \log_b(a)$ $\log_a(x) = \ln(x) / \ln(a)$ $\log_e(x) = \ln(x)$ $\log_a(a^x) = x$

$\log_{a^x}(b^x) = \log(b) / \log(a) = \log_a(b)$ $\log(b^x) / \log(a^x) = \log(b) / \log(a) = \log_a(b)$ --> x doesn't care!

$\log_2 0.125 = -3$ $\log_2 0.25 = -2$ $\log_2 0.5 = -1$ $\log_2 1 = 0$ $\log_2 2 = 1$ $\log_2 4 = 2$ $\log_2 8 = 3$ $\log_2 256 = 8$ $\log_2 65536 = 16$

$\log_{10} 0.001 = -3$ $\log_{10} 0.01 = -2$ $\log_{10} 0.1 = -1$ $\log_{10} 1 = 0$ $\log_{10} 10 = 1$ $\log_{10} 100 = 2$ $\log_{10} 1000 = 3$

$\ln(1 / e^3) = -3$ $\ln(1 / e^2) = -2$ $\ln(1 / e) = -1$ $\ln(1) = 0$ $\ln(e) = 1$ $\ln(e^2) = 2$ $\ln(e^3) = 3$

$e^{\ln(0.1)} \approx e^{-2.303} \approx 0.1$ $e^{-1} \approx 0.3679$ $e^0 = 1$ $e^1 = e \approx 2.718$ $e^2 \approx 7.389$ $e^{\ln(10)} \approx e^{2.303} \approx 10$

$$\lim_{n \to \infty} (1 + 1/n)^n = e$$

14.4 Imaginary Numbers

$i^2 = -1$ $e^{i\pi} = i^2 = -1$ $\ln(i) = i \cdot \pi / 2$ $e^{i\theta} = \cos(\theta) + i \sin(\theta)$

14.5 Trigonometric Functions

$\sin^2(x) + \cos^2(x) = 1$ $\sin(30°) = \cos(60°) = 0.5$ $\sinh(x) = (e^x - e^{-x}) / 2$ $\cosh(x) = (e^x + e^{-x}) / 2$

14.6 Difference quotients

See also: https://en.wikipedia.org/wiki/Difference_quotient
The german version of this Wikipedia page is much better: https://de.wikipedia.org/wiki/Differenzenquotient
See also: https://en.wikipedia.org/wiki/Finite_difference_coefficient

$$\frac{dw}{dx} \approx \frac{w_{n+1} - w_n}{\Delta x} \quad \text{or} \quad \approx \frac{w_n - w_{n-1}}{\Delta x} \quad \text{or} \quad \approx \frac{w_{n+1} - w_{n-1}}{2\Delta x}$$

$$\frac{d^2w}{dx^2} \approx \frac{w_{n-1} - 2w_n + w_{n+1}}{\Delta x^2}$$

$$\frac{d^3w}{dx^3} \approx \frac{-w_{n-1} + 3w_n - 3w_{n+1} + w_{n+2}}{\Delta x^3} \approx \frac{-w_{n-2} + 2w_{n-1} - 2w_{n+1} + w_{n+2}}{2\Delta x^3}$$

$$\frac{d^4w}{dx^4} \approx \frac{w_{n-2} - 4w_{n-1} + 6w_n - 4w_{n+1} + w_{n+2}}{\Delta x^4}$$

14.7 Approximations

$1 / (1 + a) \approx 1 - a$ For small a

$e \approx 2.718281828...$ $\pi \approx 3.141592654...$

$\pi \approx \sqrt[4]{9^2 + 19^2 / 22} = 3.141592653...$

Approximation for imperfect squares:

$\sqrt{n} \approx (n + a) / (2 \cdot \sqrt{a})$

where a is the nearest perfect square root number.

Example:

$\sqrt{23} \approx (23 + 25) / (2 \cdot \sqrt{25}) = 4.8$

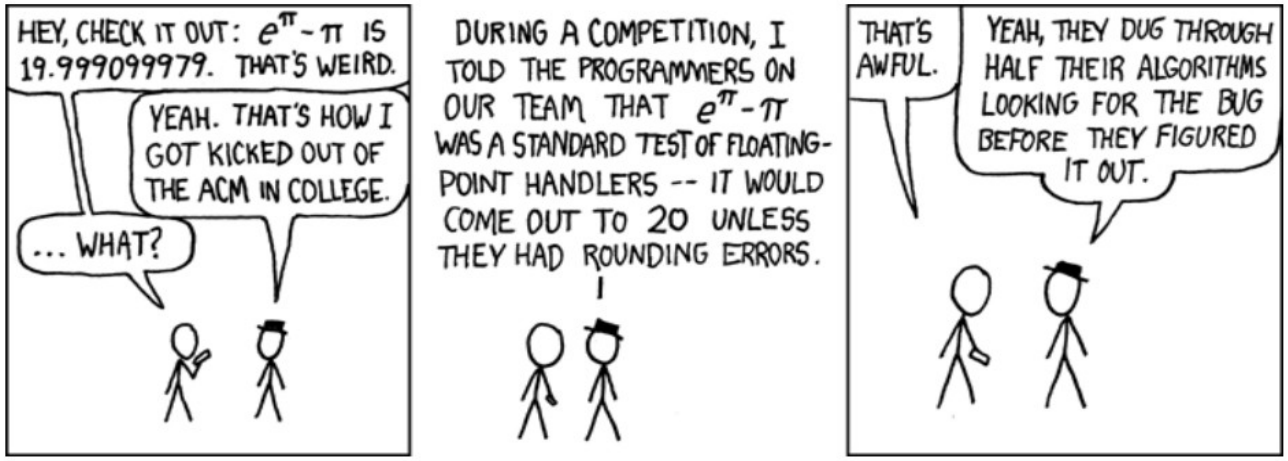
$\ln x \approx 3(x^2-1) / ((x+1)^2 + 2x)$ Source: https://www.johndcook.com/blog/2024/02/24/log-approximation/

x	$\ln(x)$	$\frac{3(x^2-1)}{(x+1)^2 + 2x}$		$2\left(\frac{x-1}{x+1}\right)$		$2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3}\right)$		$2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5}\right)$	
		Value	Error	Value	Error	Value	Error	Value	Error
0.0183	-4	-2.7935	-30%	-1.9281	-51.8%	-2.5253	-36.9%	-2.8584	-28.5%
0.0498	-3	-2.4903	-17%	-1.8102	-39.7%	-2.3047	-23.2%	-2.5477	-15.1%
0.1353	-2	-1.8885	-5.6%	-1.5233	-23.8%	-1.8177	-9.1%	-1.9202	-4.0%
0.3679	-1	-0.9950	-0.5%	-0.9242	-7.6%	-0.9900	-1.0%	-0.9985	-0.2%
1	0	0	0	0	0	0	0	0	0
2.7183	1	0.9951	-0.5%	0.9242	-7.6%	0.9900	-1.0%	-0.9985	-0.2%
7.3891	2	1.8883	-5.6%	1.5232	-23.8%	1.8177	-9.1%	1.9202	-4.0%
20.086	3	2.4904	-17%	1.8103	-39.7%	2.3047	-23.2%	2.5477	-15.1%
54.598	4	2.7934	-30%	1.9281	-51.8%	2.5253	-36.9%	2.8584	-28.5%

See also: https://www.mathematik.ch/anwendungenmath/logarithmen/

x	y = 0.2 * ln(x)	
exp(-5) = 0.006738	-1.0	
exp(-4) = 0.018316	-0.8	
exp(-3) = 0.049787	-0.6	
exp(-2) = 0.135335	-0.4	
exp(-1) = 0.367879	-0.2	
exp(-0) = 1	0	

$e^{\pi} - \pi = 19.99909998... \approx 20$ https://xkcd.com/217/



14.8 Exact Solutions

10! seconds = 6 weeks

14.9 Differential Equations

Ordinary differential equation (ODE) https://en.wikipedia.org/wiki/Ordinary_differential_equation

Partial differential equation (PDE) https://en.wikipedia.org/wiki/Partial_differential_equation

14.10 Pizza Slices Packing

Smallest Known Packing of n Sectors of 1/m th of a Unit Circle:

https://erich-friedman.github.io/mathmagic/0112.html

https://erich-friedman.github.io/mathmagic/

14.11 Geometry

Right triangle: $a^2 + b^2 = c^2$ $h^2 = p \cdot q$

14.12 Divisibility

Divisibility rules: https://en.wikipedia.org/wiki/Divisibility_rule

14.13 Integration Examples

$$\int_1^2 \int_2^3 \int_0^1 8xyz \, dz \, dx \, dy$$

$$= \int_1^2 \int_2^3 4xyz^2 \Big]_0^1 \, dx \, dy$$

$$= \int_1^2 \int_2^3 \left[(4xy1^2) - (4xy0^2)\right] dx \, dy$$

$$= \int_1^2 \int_2^3 (4xy1^2) \, dx \, dy$$

$$= \int_1^2 \int_2^3 4xy \, dx \, dy$$

$$= \int_1^2 2x^2y \Big]_2^3 \, dy$$

$$= \int_1^2 \left[(2(3)^2y) - (2(2)^2y)\right] dy$$

$$= \int_1^2 10y \, dy$$

$$= 5y^2 \Big]_1^2 \, dy$$

$$= \left[5(2)^2 - 5(1)^2\right]$$

$$= 15$$

14.14 CAS (Computer Algebra Systems)

Wolfram Alpha: https://www.wolframalpha.com/

Syntax examples: "series of y=0.006541*2^(10*x) at x=0.421"

GeoGebra: Can be used online: https://www.geogebra.org/cas or as a standalone application: https://www.geogebra.org/download

https://wiki.geogebra.org/en/PointList_Command	**Creates list of points from a list of two-element lists. Example: PointList({{1,2},{3,4}})**
https://wiki.geogebra.org/en/FitPoly_Command	**Calculates the regression polynomial of degree *n*. Example: FitPoly({(-1, -1), (0, 1), (1, 1), (2, 5)}, 3)**
https://wiki.geogebra.org/en/FitExp_Command	**Calculates the exponential regression curve in the form $a e^{bx}$.**
https://wiki.geogebra.org/en/FitLog_Command	**Calculates the logarithmic regression curve a + b ln(x).**
https://wiki.geogebra.org/en/FitGrowth_Command	**Calculates a function of the form a b^x to the points in the list.**
https://wiki.geogebra.org/en/FitPow_Command	**Calculates the regression curve in the form $a\ x^b$.**
https://wiki.geogebra.org/en/FitSin_Command	**Calculates the regression curve in the form *a + b sin (c x + d)*.**
https://wiki.geogebra.org/en/FitLine_Command	**Calculates the *y* on *x* regression line of the points.**
https://wiki.geogebra.org/en/FitLineX_Command	**Calculates the *x* on *y* regression line of the points.**
https://wiki.geogebra.org/en/FitLogistic_Command	**Calculates the regression curve in the form a / (1 + b e^(-kx)).**
https://wiki.geogebra.org/en/Fit_Command	**Calculates a linear combination of the *functions* that best fit the *points* in the list.**

14.15 Method of Adjoint Systems, Reciprocity

See also: Laning, J. H. and Battin R. H.: Random Processes in Automatic Control (Control Systems Engineering S.), page 239

Reciprocity:

In electric circuit theory it is well known that a current in one loop, say loop 1, of a network caused by a voltage source in any other loop, say loop 2, of the same network is identical to the current which would be produced in loop 2 by the same voltage source in loop 1. In other words an ideal volage source and an ideal ammeter may be interchanged without altering the reading of the instrument.

As a second example of reciprocity consider a tightly stretched string . Then a concentrated load applied at position 1 of the string produces the same deflection at position 2 as would be produced at position 1 by the same load applied at position 2.

See also: https://en.wikipedia.org/wiki/Reciprocity_(electrical_networks)

See also: https://en.wikipedia.org/wiki/Reciprocity_(engineering)

See also: https://en.wikipedia.org/wiki/Reciprocity_(electromagnetism)

15 SRS SIM900

SIM = Small Instrumentation Modules

https://www.thinksrs.com/products/sim.html

SIM900 Mainframe Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM900m.pdf

PuTTY Terminal program: https://www.putty.org/

DIP Switches for 9600 Baud: 7 on, all others off.

Example commands (case-insensitive) and responses from SIM900, with SIM921 in slots 1+2 and SIM965 in slot 3:

*rst	(no reply, this is the reset command, normally not required after power-on)
*idn?	Stanford_Research_Systems,SIM900,s/n072269,ver3.5
conn 1,"xxx"	(no reply, 1 is the slot number and xxx is the exit string)
*idn?	Stanford_Research_Systems,SIM921,s/n009216,ver3.6
xxx	(no reply, this is the exit string)
*idn?	Stanford_Research_Systems,SIM900,s/n072269,ver3.5
conn 3,"xxx"	(no reply, 3 is the slot number and xxx is the exit string)
*idn?	Stanford_Research_Systems,SIM965,s/n005140,ver3.8

15.1 SIM910 & SIM911 JFET & BJT Preamplifiers

https://www.thinksrs.com/products/sim910911.html

SIM910 JFET Preamplifier Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM910m.pdf

SIM911 BJT Preamplifier Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM911m.pdf

15.2 SIM921 AC Resistance Bridge

https://www.thinksrs.com/products/sim921.html

SIM921 AC Resistance Bridge Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM921m.pdf

I have one SIM921, but I'm interested to buy one more. I you have or see a SIM921 for sale, please let me know.

The module is mostly self-explanatory, but setting the DISPLAY needs to be explained. Directly beneath [Set] is the DISPLAY block of the panel. The two buttons at the bottom of this section, [Display], select the quantity for display in the numeric field. The selections are:

Value	The measured value of the user's resistor-under-test is displayed, either in resistance units or temperature units (depending on the units selection, below). When Value is displayed, pressing [Set] acts as a short-cut to reset the output filter. This can be useful to speed settling with a long time constant after a large resistance change is made, or after the range or excitation is changed.
Value−Offset	This selection (also known as "deviation") also displays the measurement result (either in resistance or temperature units), but after subtracting the user-settable Offset. Pressing [Set] will reset the output filter.
Phase (deg.)	This selection shows the phase angle between measured current and voltage (in degrees), and is an indication of how much capacitive loading is present. Phase is positive for capacitive loads. A phase angle near +90◦ should be viewed with caution: this indicates that most of the current is flowing through the reactive part of the load, and measurement accuracy may suffer. When phase is displayed, [Set] can be used to modify the model used in the SIM921 to determine resistance. By pressing [Set], the numeric display will show the word: Zero This forces the meter to assume the phase angle between the voltage and current is zero when solving for R. This is helpful when measuring very small resistances (such as superconducting samples), since the phase determination becomes otherwise ill-conditioned when the voltage signal approaches zero, and causes excess noise in the results. Pressing [Set] restores normal operation.
Offset	The offset, or setpoint, is the user-selected value to subtract from the sensor measurement. The offset is used in the Value−Offset display (above), as well as to determine the analog output voltage (see below). The [Set] buttons will accelerate through multiple digits to adjust the offset; two short cuts also exist. If both [Set] and [Set] are pressed simultaneously, Offset is preloaded with the latest measurement result of Value. Depressing both buttons again will force Offset to zero.
Freq. (Hz)	This field controls the excitation frequency for the SIM921. [Set] adjusts the frequency from 2 Hz to 60 Hz. Depressing both [Set] and [Set] together will step between 15 Hz, 10 Hz, 5 Hz, and 2.5 Hz.

AOUT	**This parameter is the slope (in V/Ω or V/K) used to scale the deviation signal for analog output. Use [Set] to accelerate through many orders of magnitude for AOUT; releasing the button and re-pressing it allows fine control over the lower digits, as the setting begins accelerating again. If resistance units are selected for analog output, the Ω indicator will be lit next to the numeric display; if temperature units are selected, the K indicator will be lit.**
Units (Ω, K)	**This is actually three separate selections that are stepped through by continuing to press [Display]. The first selection lights both the Value and Units indicators. This selects either resistance or temperature units for the Value display.** **Use [Set] to switch between resistance (the display will show) and temperature (the display will show the ID message of the selected sensor calibration curve).** **Pressing [Display] again will light AOUT and Units together. Now, [Set] selects between resistance or temperature units for the analog output function. Note that the deviation display and offset parameter units are also determined by AOUT–Units.** **Pressing [Display] one final time will leave Units lit alone. Now the [Set] selects among three sensor calibration curves stored in the SIM921. If a particular curve has not been loaded, the is lit to indicate this is not a usable curve; once (at least) two points are loaded in a sensor curve memory, the display will show to the left of the curve ID. Only one curve can be selected at a time.**

Example commands and responses from SIM900, with SIM921 in slots 1+2: (tested with PuTTY)

*idn?	Stanford_Research_Systems,SIM900,s/n072269,ver3.5
lcme?	(show last error code)
conn 1,"xxx"	(no reply, 1 is the slot number and xxx is the exit string)
*idn?	Stanford_Research_Systems,SIM921,s/n009216,ver3.6
freq?	9.9994 (Frequency in Hz)
rang?	5
exci?	8
exon?	1
mode?	0
iexc?	+3.021547E-05 30.2 µA
vexc?	+3.320710E-02 33.2 mV
rval?	+1.098963E+03 1.1 kΩ
curv?	1
cini? 1	0,CU_1,0 (linear, identification string "CU_1", 0 points)
cini 1,0,PT100	
cini? 1	0,PT100,0
capt 1,80.31,223.15	PT100 -50°C
capt 1,100,273.15	PT100 0°C
capt 1,119.40,323.15	PT100 +50°C
cini? 1	0,PT100,3 These calibration points are automatically saved in NVRAM

C# Program for writing the calibration curves for PT100 (-50°C to +250°C) and B57019-K992 (-55°C to +250°C) into the SIM921 module:

```
using System;
using System.Windows.Forms;

namespace SIM900_SIM921
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            double c,k,r;
            double[] B57019 = { 46.95,    34.75,    25.97,    19.58,    14.90,   11.43,   8.845,   6.897,
                                5.419,    4.288,    3.418,    2.742,    2.215,   1.800,   1.471,   1.210,
                                1.000,    0.8312,   0.6944,   0.5831,   0.4919,  0.4168,  0.3548,  0.3033,
                                0.2604,   0.2244,   0.1941,   0.1685,   0.1468,  0.1283,  0.1126,  0.09906,
                                0.08744,  0.07741,  0.06873,  0.06119,  0.05462, 0.04889, 0.04387, 0.03945,
                                0.03557,  0.03214,  0.02911,  0.02642,  0.02403, 0.02190, 0.01999, 0.01829,
                                0.01676,  0.01539,  0.01415,  0.01304,  0.01203, 0.01112, 0.01029, 0.009540,
                                0.008856, 0.008233, 0.007665, 0.007145, 0.00669, 0.006233 };

            InitializeComponent();

            serialPort1.BaudRate = 9600;
            serialPort1.DataBits = 8;
            serialPort1.Parity = System.IO.Ports.Parity.None;
            serialPort1.StopBits = System.IO.Ports.StopBits.One;
            serialPort1.Handshake = System.IO.Ports.Handshake.XOnXOff;
            serialPort1.PortName = "COM1";
            serialPort1.NewLine = "\r\n";                // CR LF
            serialPort1.Open();
            serialPort1.WriteLine("xxx");                // Escape string
            System.Threading.Thread.Sleep(100);
            serialPort1.WriteLine("*RST");               // SIM900 reset
            System.Threading.Thread.Sleep(100);
            serialPort1.WriteLine("CONN 1,\"xxx\"");     // SIM921 in Slot 1
            System.Threading.Thread.Sleep(100);
            serialPort1.WriteLine("CINI 1,0,PT100");     // Curve 1 is for PT100
            System.Threading.Thread.Sleep(100);

            for (int n = 0; n < 151; n++)                // PT100, 151 points calculated by formula
            {
                c = -50 + 2 * n;    // -50°C to +250°C    this is PTC, sorted for increasing resistance
```

```
                k = 273.15 + c;     // Kelvin
                if (c < 0)
                    r = 100 * (1 + 3.9083e-3 * c - 5.775e-7 * c * c - 4.183e-12 * c * c * c * (c - 100));
                else
                    r = 100 * (1 + 3.9083e-3 * c - 5.775e-7 * c * c);
                richTextBox1.AppendText(c + "   " + r.ToString("F5") + "   \n");
                serialPort1.WriteLine("CAPT 1," +
                                      r.ToString("F5",System.Globalization.NumberFormatInfo.InvariantInfo) +
                                      "," +
                                      k.ToString("F5",System.Globalization.NumberFormatInfo.InvariantInfo));
                System.Threading.Thread.Sleep(100);    // this delay seems to be necessary !!!
            }

            serialPort1.WriteLine("CINI 2,0,B57019-K992");     // Curve 2 is for B57019-K992
            System.Threading.Thread.Sleep(100);

            for (int n = 61; n >= 0; n--)                // B57019-K992, 62 Points from "1101" curve in datasheet
            {
                c = -55 + 5 * n;    // +250°C to -55°C    this is NTC, sorted for increasing resistance
                k = 273.15 + c;     // Kelvin
                r = 9921.0 * B57019[n];

                richTextBox1.AppendText(c + "   " + r.ToString("F5") + "   \n");
                serialPort1.WriteLine("CAPT 2," +
                                      r.ToString("F5", System.Globalization.NumberFormatInfo.InvariantInfo) +
                                      "," +
                                      k.ToString("F5", System.Globalization.NumberFormatInfo.InvariantInfo));
                System.Threading.Thread.Sleep(100);    // this delay seems to be necessary !!!
            }

            serialPort1.WriteLine("xxx");                // Escape string
            System.Threading.Thread.Sleep(100);
        }
    }
}
```

15.2.1 PT100

PT100 Table: https://www.temperaturmesstechnik.de/fileadmin/user_upload/pdf/tmh_pt100_tabelle.pdf
https://en.wikipedia.org/wiki/Resistance_thermometer

PT100 Datasheet from Jumo: https://www.jumo.net/attachments/JUMO/attachmentdownload?id=1222871

Dimensions 2mm x 10mm, Self heating is about 0.15K/mW, Time constants in air are $t_{0.5}$ = 3s and $t_{0.9}$ = 10s
With the maximum excitation voltage of the SIM921 (30 mV) the power at 100 Ω is $P = U^2 / R$ = 9 µW so that self heating is negligible.

15.2.2 Ultra-Small NTCs for fast measurement of air temperature:

Siemens NTC B57019-K992, 0.4 mm x 0.3 mm, time constant in air 0.4 s, https://www.datasheetarchive.com/datasheet?id=56ead9774ec2a479df218d62034b1540d4ac80&type=M&term=b57019k992k

If anybody knows a source for good old B57019 sensors, please let me know.

TDK NTC B57540G1103G000, diameter 0.8 mm, height 1.4 mm, self heating 0.4 mW/K, time constant in air about 3 s,

https://product.tdk.com/system/files/dam/doc/product/sensor/ntc/ntc_element/data_sheet/50/db/ntc/ntc_glass_enc_sensors_g1540.pdf

Small 0.4mm x 0.2mm NTCs from Murata:
https://www.murata.com/-/media/webrenewal/products/thermistor/ntc/ncp/ncp02.ashx?la=en-gb&cvid=20240717024027000000
https://www.murata.com/-/media/webrenewal/products/thermistor/ntc/ncp/ncp02.ashx?la=en-us&cvid=20231225010000000000
https://www.win-source.net/products/detail/murata-electronics-north-america/ncp02xh103f05rh.html

Small 0.4mm x 0.2mm NTCs from TDK: https://product.tdk.com/system/files/dam/doc/product/sensor/ntc/chip-ntc-thermistor/catalog/tpd_commercial_ntc-thermistor_ntcg_en.pdf

Small 0.6mm x 0.3mm NTC's from Murata: https://www.murata.com/en-eu/products/thermistor/ntc/overview/lineup/ncu

See also: https://www.edn.com/contactless-electric-bell-on-a-gradient-relay/

Imperial Code	Metric Code	Length [mm]	Width [mm]
01005	0402	0.4	0.2
0201	0603	0.6	0.3
0402	1005	1.0	0.5
0603	1608	1.6	0.8
0805	2012	2.0	1.2
1206	3216	3.2	1.6

15.3 SIM925 Octal Four-Wire Multiplexer

https://www.thinksrs.com/products/sim925.html

SIM925 Octal Four-Wire Multiplexer Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM925m.pdf

I need a SIM925. If you want to sell one, plese let me know.

15.4 SIM960 Analog PID Controller

https://www.thinksrs.com/products/sim960.html

SIM960 Analog PID Controller Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM960m.pdf

I have a few of these in stock, please contact me if you need one.

15.5 SIM965 Bessel & Butterworth filters

https://www.thinksrs.com/products/sim965.html

SIM965 Bessel & Butterworth Filter Manual: https://www.thinksrs.com/downloads/pdfs/manuals/SIM965m.pdf

16 Miscellaneous

16.1 Power Supply

The supply voltage of THAT is 5V, coming from a USB port. The current is about 120mA without any connections, or about 240mA with heavy load at two outputs (for example in the circuit with the resolver).

There is a BD2244G-M current limiter at the USB port, which limits the current to 590mA. It doesn't have a protection against wrong polarity. The absolute maximum rating of the input voltage is -0.3V to +7.0V

Data sheet: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/power_switch/bd2244g-m-e.pdf

THAT schematic: https://github.com/anabrid/the-analog-thing/blob/main/ANATHING_OSH/ANATHING-BASE%20-%20Schematic.pdf

16.2 Find suitable parallel or serial Combinations of Resistors

https://www.electronicdeveloper.de/widerstandaus2r.aspx

Voltage divider for making 2.048 V from 10 V: 10.0 kΩ + 2575.45 kΩ 2575.45 kΩ ≈ 2.7 kΩ || 56 kΩ

16.3 The Physics of Hula-Hoop

https://www.pnas.org/doi/10.1073/pnas.2411588121

16.4 Memristor and Corristor

https://corristor.com **https://techifab.com/**

https://www.kickstarter.com/projects/corristor/cor-ristor-your-next-level-correlation-buddy

german: https://de.wikipedia.org/wiki/Memristor
english: https://en.wikipedia.org/wiki/Memristor

Data sheet: https://techifab.com/wp-content/uploads/2024/07/2024-07-16_Product_specification_COR-RiSTOR.pdf

A simple memristor model: https://analogparadigm.com/downloads/alpaca_48.pdf

Paper in Journal of Applied Physics: https://pubs.aip.org/aip/jap/article/135/20/200902/3295370/Prospects-for-memristors-with-hysteretic

Memristance = Voltage Momentum / Current Momentum **Voltage Momentum = Time-integrated Voltage** **Current Momentum = Charge ?**

ICT = Information Communications Technology

SHT401-HD1B Sensor for relative humidity and temperature:
https://sensirion.com/media/documents/74DC2069/661CD1A9/HT_DS_Datasheet_SHT4xI-analog.pdf
Temperature in °C = -66.875 + 43.75 * V where V is the sensor output voltage in V
Relative humidity in % = -12.5 + 25 * V where V is the sensor output voltage in V
The S2 switch selects temperature (side with "S2" label) or humidity (side near J3 connector).

How to modify the sensor board for simulataneously measuring temperature and humidity:

- **Add a second RJ45 connector to the board.**
- **Connect pins 2, 7 and 8 to the ground plane**
- **Connect pin 1 to the pin of the switch, which is most far away from the edge of the board. This is the humidity signal without amplification.**

Turn the potentiometer on the sensor boards fully counter-clockwise for gain = 1. The hardware gain must be the same as the gain that's set in software. In the example, the gain was set to 1.11 because the sensor's output signal goes up to 4.5V, which after amplification by factor 1.11 becomes 5V, which seems to be the maximum input voltage of the board.

AMS6916-1200-B-H Sensor for barometric air pressure 700 - 1200 mbar:
https://www.analog-micro.com/products/pressure-sensors/board-mount-pressure-sensors/ams6916/ams6916-datasheet.pdf
Pressure in mbar = 637.5 + 125 * V where V is the sensor output voltage in V
The air pressure in weather forecasts is always for 0 m altitude. The air pressure decreases by about 1 mbar every 8 m.

Altitude	0 m	100 m	200 m	250 m	300 m
Air pressure	+-0 mbar	-12.5 mbar	-25.0 mbar	-31.25 mbar	-37.5 mbar

Pinout of the RJ45 sensor connectors:

1	Signal The input impedance of the main board depends on the number of channels this signal is connected to: Channels: 1 | 2 | 3 | 4 | 5 | 6 Input impedance: 15 kΩ | 7.5 kΩ | 5 kΩ | 3.75 kΩ | 3 kΩ | 2.5 kΩ
2	GND
3	unknown
4	+6V (supply voltage for OP-Amp) Add a 100nF capacitor between pins 2 and 4
5	+5V (supply voltage for sensors) Add a 100nF capacitor between pins 5 and 7 According to the datasheet this should be 3.3V, but it's 5V. https://techifab.com/wp-content/uploads/2024/07/2024-07-16_Product_specification_COR-RiSTOR.pdf
6	unknown
7	GND
8	GND

The sample interval seems to be about 1.5 seconds and can't be changed.

More help is found here: https://techifab.com/help/

Open questions:

- **Instructions for a demo project?**
- **How can I generate the "target" column in the data file?**
- **Which role does the memristor play in the correlation analysis? It seems to be very well hidden in a black box. What does the memristor do, that couldn't also be done by a digital computer?**

16.5 Hole through Earth

HOLE THROUGH THE EARTH

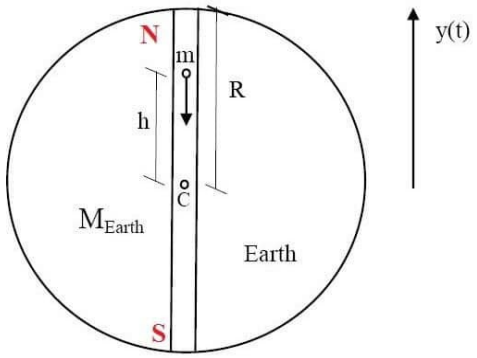


Fig. 1: Mass m falling along a hole drilled through the Earth.

$F = G\frac{mM_h}{h^2} = G\frac{m[\frac{4\pi h^3}{3}\rho]}{h^2} = \frac{4}{3}\pi G m\rho h = -ma$, where: $M_h = \frac{4\pi h^3}{3}\rho$ is the mass of the earth under the falling mass m. So:

$$a = -\frac{4}{3}\pi G\rho h \qquad (1.1)$$

Therefore, the falling mass m makes a harmonic motion; in fact, from the (1.1), we have:

$ma = m\frac{d^2y}{dt^2} = -\frac{4}{3}\pi Gm\rho y$, from which:

$$m\frac{d^2y}{dt^2} + \frac{4}{3}\pi Gm\rho y = 0 \ , \qquad (1.2)$$

whose solution is: $y = R\cos\omega t$ (which, by substitution, satisfies the (1.2) indeed, provided that: $\omega = \sqrt{\frac{4\pi G\rho}{3}}$). If the Earth density is $\rho = 5.5\cdot 10^3 kg/m^3$, we have: $T = \frac{2\pi}{\omega} = 5.06\cdot 10^3 s \cong 84\,\text{min}$ (to and fro); from pole to pole it's 42 min. About the max speed at the centre of the Earth, ($R = 6.3715\cdot 10^6 m$): $V_{Max} = \omega R = 7.9 km/s$.

16.6 Printed Circuit Boards

https://www.seeedstudio.com/fusion_pcb.html

Seeed bietet hier: http://support.seeedstudio.com/knowledgebase/articles/1176949-how-to-generate-gerber-and-drill-files-from-eagle, eine sehr gute Beschreibung an, auch links zum Herunterladen der CAM-Prozessordaten. Zur Kontrolle kannst du deine Gerberdaten bzw. das Ergebnis auf der SEEED Seite betrachten. Geht super!

17 Datasheets

Analog Devices:

Multipliers and Dividers: https://www.analog.com/en/parametricsearch/11106#/

Logarithmic Transimpedance Amplifiers: https://www.analog.com/en/parametricsearch/10772#/

AD534: $[(X1 - X 2) \cdot (Y1 - Y 2) / 10\ V] + Z\ 2$ https://www.analog.com/media/en/technical-documentation/data-sheets/AD534.pdf

AD538: $V_O = V_Y \cdot (V_Z / V_X)^M$ https://www.analog.com/media/en/technical-documentation/data-sheets/AD538.pdf

AD633: $W = (X1 - X2) \cdot (Y1 - Y2) / 10V + Z$ https://www.analog.com/media/en/technical-documentation/data-sheets/AD633.pdf

AD639: Sin, Cos, Tan... https://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/AD639.pdf

AD734: This multiplier is better than AD633, but very expensive. https://www.analog.com/media/en/technical-documentation/data-sheets/AD734.pdf

AD818: Paired transistors for log / antilog Note: The same type number is also used for a video amplifier chip, which is a completely different thing!
https://www.analog.com/media/en/analog-dialogue/volume-6/number-3/articles/volume6-number3.pdf

BD2244G-M This current limiter is used at the USB power input port, limiting the current to 590mA.

Data sheet: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/power_switch/bd2244g-m-e.pdf

Texas Instruments / Burr-Brown:

LOG112, LOG2112, Precision Logarithmic and Log Ratio Amplifiers: https://www.ti.com/lit/ds/symlink/log112.pdf

17.1 Op-Amps

Type	Supply Voltage	Offset Voltage		Output Current	Slew Rate	Package	Link to Datasheet, Notes
		Typical	Maximum				
AD549J	+-15 V	+-0.5 mV	+-1 mV		2 V/µs	TO-99	https://www.analog.com/media/en/technical-documentation/data-sheets/AD549.pdf Ultra-low bias current Warning: Counterfeit chips exist
AD549K		+-0.15 mV	+-0.5 mV				
AD549L		+-0.3 mV	+-0.5 mV				
AD549S		+-0.3 mV	+-0.5 mV				
AD704	+-2 V to +-18 V	+-0.05 mV	+-0.15 mV	15 mA	0.1 V/µs	DIP, SO	https://www.analog.com/media/en/technical-documentation/data-sheets/AD704.pdf
AD711	+-4.5 V to +-18 V	+-0.3 mV	+-2 mV	25 mA	16 V/µs	DIP8, SO8	https://www.analog.com/media/en/technical-documentation/data-sheets/ad711.pdf
AD8531, AD8532, AD8534	2.7 V to 6.0 V	no spec	+-25mV	250 mA	5 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/AD8531_8532_8534.pdf Rail-to-Rail I/O
AD8541, AD8542, AD8544	2.7 V to 5.5 V	+-1 mV	+-6 mV	15 mA	0.75 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/AD8541_8542_8544.pdf Rail-to-Rail I/O https://www.reichelt.de/operationsverstaerker-2-fach-0-92-v-s-1-mhz-so-8-ad-8542-arz-p185453.html
AD8601A, AD8602A, AD8604A	2.7 V to 5.5 V	+-0.08 mV	+-0.5 mV	30 mA	5 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/ad8601_8602_8604.pdf Rail-to-Rail I/O https://www.reichelt.de/index.html?ACTION=446&LA=446&nbc=1&q=ad8602
AD8601D, AD8602D, AD8604D		+-1.3 mV	+-6 mV				
AD8628, AD8629, AD8630	2.7 V to 5.0 V	+-0.001 mV	+-0.005 mV	30 mA	1 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/ad8628-8629-8630.pdf Rail-to-Rail I/O, Ultra-low offset voltage https://www.reichelt.de/operationsverstaerker-2-fach-1-v-s-2-5-mhz-msop-8-ad-8629-armz-p185470.html
ADA4510-2	+-3 V to +-20 V	+-0.005 mV	+- 0.02 mV	22 mA	19 V/µs	SO8	https://www.analog.com/media/en/technical-documentation/data-sheets/ada4510-2.pdf Rail-to-Rail I/O, Ultra-low offset voltage
LMH6639	3 V to +-5 V	1.03 mV	5 mV	110 mA	200 V/µs		https://www.ti.com/lit/ds/symlink/lmh6639.pdf Rail-to-Rail Output

LT1014	to +-22 V	+-0.05 mV	+- 0.18 mV		0.4 V/µs	DIP, SO	https://www.analog.com/media/en/technical-documentation/data-sheets/LT1013-LT1014.pdf
LTC1051, LTC1053	4.75 V to 16 V	+-0.0005 mV	+-0.005 mV		4 V/µs	DIP	https://www.analog.com/media/en/technical-documentation/data-sheets/10513fa.pdf Ultra-low offset voltage, zero drift
LT1356	to +-18 V	+-0.3 mV	+-0.8 mV	30 mA	400 V/µs	SO, DIP	https://www.analog.com/media/en/technical-documentation/data-sheets/13556fc.pdf
MCP606/7/8/9	2.5 V to 6 V		+-0.25 mV	17 mA	0.08 V/µs		https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/11177f.pdf Rail-to-Rail output
MCP6021/1R/2/3/4	2.5 V to 5.5 V		+-0.5 mV	22 mA	7 V/µs		https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP6021-Data-Sheet-DS20001685.pdf Rail-to-Rail I/O
OP07C	6 V to 36 V	+-0.06 mV		?	0.3 V/µs		https://www.ti.com/lit/ds/symlink/op07.pdf
OP07D		+-0.15 mV					
OP400A	to +-20 V	+-0.04 mV	+-0.15 mV	28 mA	0.15 V/µs	SO, DIP	https://www.analog.com/media/en/technical-documentation/data-sheets/op400.pdf
OP490	+-0.8 V to +-18 V	+-0.6 mV	+-1 mV		0.012 V/µs	DIP, SO	https://www.analog.com/media/en/technical-documentation/data-sheets/OP490.pdf
OP495	+-1.5 V to +-18 V	+-0.03 mV	+-0.3 mV	18 mA	0.03 V/µs	DIP, SO	https://www.analog.com/media/en/technical-documentation/data-sheets/OP295_495.pdf
OPAx192	to +-18 V	+-0.005 mV	+-0.025 mV	65 mA	20 V/µs	SO	https://www.ti.com/lit/ds/symlink/opa192.pdf Ultra-low offset voltage, high slew rate very expensive
OPA207	+-2.25 V to +-18 V	+-0.015 mV	+-0.1 mV	40 mA	2.7 V/µs		https://www.ti.com/lit/ds/symlink/opa207.pdf
OPA277P, OPA277U	+-2 V to +-18 V	+-0.01 mV	+-0.02 mV	35 mA	0.8 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa277.pdf Ultra-low offset voltage
OPA2277P, OPA2277U		+-0.01 mV	+-0.025 mV				
OPAx277PA, OPAx277UA		+-0.02 mV	+-0.05 mV				
OPA328, OPA2328 very interesting	2.2 V to 5.5 V	+-0.003 mV	+-0.05 mV	55 mA	30 V/µs		https://www.ti.com/lit/ds/symlink/opa328.pdf Rail-to-Rail I/O
OPAx354	2.5 V to 5.5 V	+-2 mV	+-8 mV	100 mA	150 V/µs		https://www.ti.com/lit/ds/symlink/opa354.pdf Rail-to-Rail I/O
OPA387, OPA2387, OPA4387 very interesting	1.7 V to 5.5 V	+-0.00025 mV	+-0.002 mV	55 mA	2.8 V/µs		https://www.ti.com/lit/ds/symlink/opa387.pdf

OPA397, OPA2397, OPA4397 very interesting	1.7 V to 5.5 V	+-0.01 mV	+-0.06 mV	55 mA	4.5 V/µs		https://www.ti.com/lit/ds/symlink/opa397.pdf Rail-to-Rail I/O
OPA445	+-10 V to +-45 V	+-1.5 mV	+-5 mV	26 mA	15 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa445.pdf
OPA454	+-5 V to +-50 V	+-0.2 mV	+-4 mV	50 mA	13 V/µs	SO	https://www.ti.com/lit/ds/symlink/opa454.pdf
OPA551, OPA552	+-4 V to +-30 V	+-1 mV	+-3 mV	200 mA	15 V/µs, 24 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa551.pdf
OPA4131 very interesting	+-4.5 V to +-18 V	+-0.2 mV	+-1 mV	20 mA	10 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa4131.pdf Ultra-low offset voltage, high slew rate very expensive
OPA4172	+-2.25 V to +-18 V	+-0.2 mV	+-1 mV	75 mA	10 V/µs	SO, TSSOP	https://www.ti.com/lit/ds/symlink/opa4172.pdf
OPA4991	2.7 V to 40 V	+-0.125 mV	+-0.83 mV	75 mA	21 V/µs	SO, TSSOP, SOT23-14	https://www.ti.com/lit/ds/symlink/opa4991.pdf
OPA4992	2.7 V to 40 V	+-0.21 mV	+-1 mV	65 mA	32 V/µs	SO, TSSOP	https://www.ti.com/lit/ds/symlink/opa4992.pdf
TL064C	+-5 V to +-15 V	+-3 mV	+-15 mV	no spec	3.5 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tl064.pdf
TL064AC, TL064I		+-3 mV	+-6 mV				
TL064BC		+-2 mV	+-3 mV				
TL071C, TL072C, TL074C	+-2.25 V to +-20V	+-3 mV	+-10 mV	26 mA	20 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tl074h.pdf TL074H is used in THAT Historically the TL07x and TL08x had different specs, but nowadays they are almost the same. TL07x had smaller noise and offset voltage. Warning: Counterfeit chips exist
TL074AC, TL074I		+-3 mV	+-6 mV				
TL074BC		+-2 mV	+-3 mV				
TL074M		+-3 mV	+-9 mV				
TL074H		+-1 mV	+-4 mV				
TL081C, TL082C, TL084C	+-2.25 V to +-20V	+-3 mV	+-15 mV	26 mA	20 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tl084h.pdf Warning: Counterfeit chips with TI logo exist TL084H is available in SOT23-14 package, which has almost the same footprint as DFN14. Adapters from DFN14 to DIP are available here: https://www.artekit.eu/products/breakout-boards/bbadapters/dfn-to-dip/dfn-14-to-dip-adapter-4mm-x-3mm-p0-50-pack-of-2/
TL084AC, TL084I		+-3 mV	+-6 mV				
TL084BC		+-2 mV SGS: +-1 mV	+-3 mV				
TL084H		+-1 mV	+-4 mV				
TLC271	3V to 16V					DIP	https://www.ti.com/lit/ds/symlink/tlc271.pdf old design, not recommended
TLC272	3V to 16V	+-1.1 mV	+-10 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc272.pdf
TLC274	3V to 16V	+-1.1 mV	+-10 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc274.pdf

TLC277	3V to 16V	+-0.2 mV	+-0.5 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc272.pdf
TLC279	3V to 16V	+-0.32 mV	+-0.9 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc274.pdf
TLC2272, TLC2274	+-2.2 V to +-8 V	+-0.3 mV	+-2.5 mV		3.6 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tlc2272.pdf Rail-to-Rail Output (but not Input)
TLC2272A, TLC2274A		+-0.3 mV	+-0.95 mV				
TLV9061	1.8V to 5.5V	+-0.3 mV	+-1.6 mV		6.5 V/µs		https://www.ti.com/lit/ds/symlink/tlv9062.pdf Rail-to-Rail I/O
TLV9062							
TLV9064							

17.1.1 Slew Rate

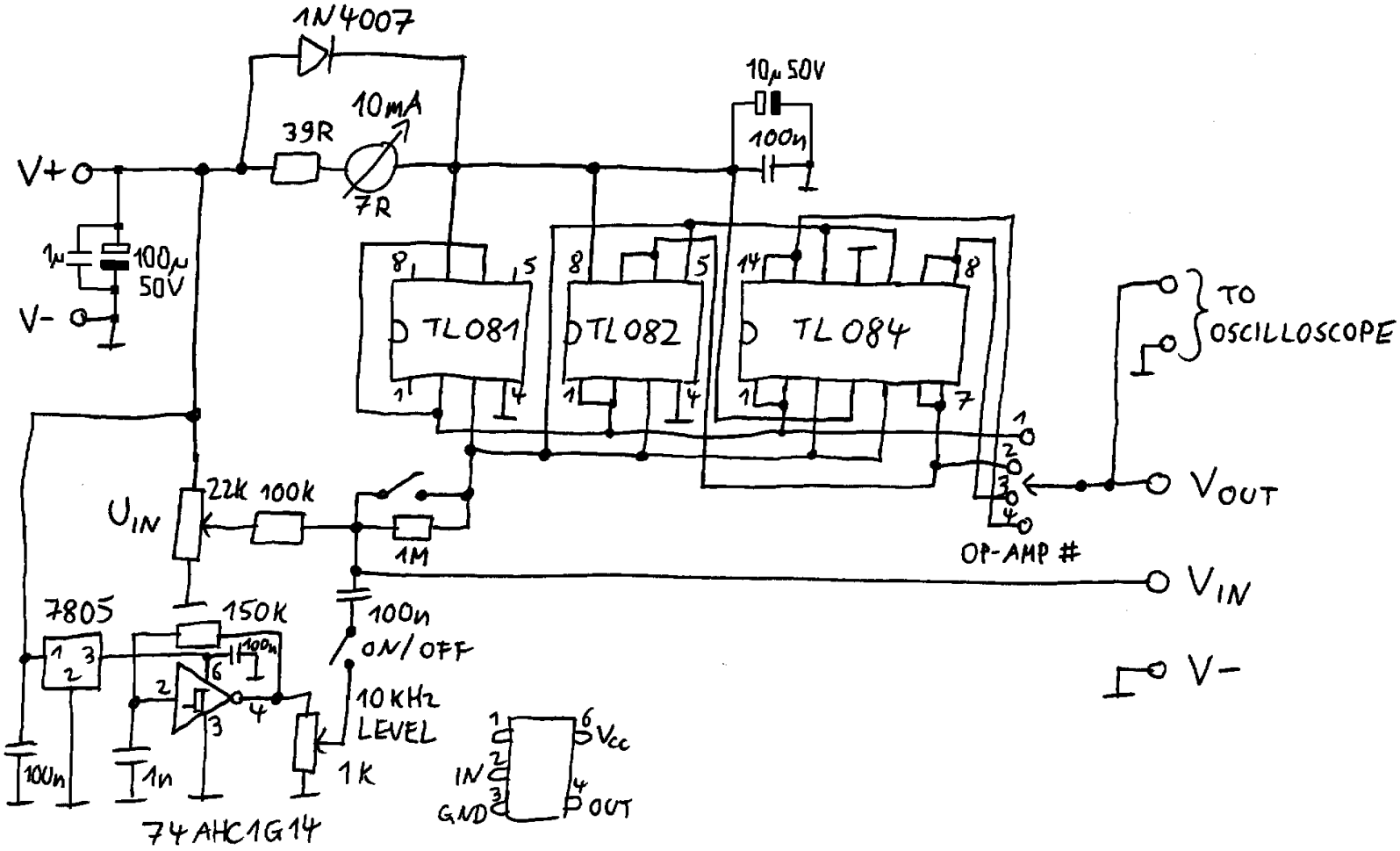
Slew Rate = A · 2 · π · f

with A = Peak amplitude of sine wave in [V] and f = Frequency of sine waveform in [Hz]

Peak Amplitude	Sine Frequency	Slew Rate
1 V	1 kHz	6.28 V/ms
10 V	1 kHz	62.8 V/ms
10 V	2 kHz	125.6 V/ms
10 V	5 kHz	314 V/ms
10 V	10 kHz	628 V/ms
10 V	20 kHz	1.26 V/µs
10 V	50 kHz	3.14 V/µs
10 V	100 kHz	6.28 V/µs
10 V	200 kHz	12.6 V/µs
10 V	500 kHz	31.4 V/µs
10 V	1 MHz	62.8 V/µs
10 V	2 MHz	126 V/µs
10 V	5 MHz	314 V/µs

17.1.2 OP-Amp Tester

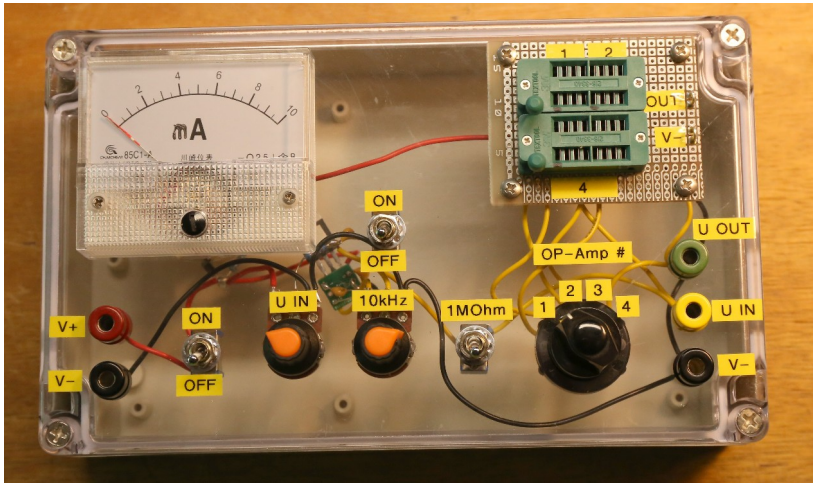
Because too many fake chips came from China, it was necessary to build an OP-Amp tester:



Don't make the supply voltage bigger than 35V, because that's the maximum input voltage of the 7805 voltage regulator.

These parameters can be tested:

Parameter:	How to test it:
Supppy current	It's shown in the mA meter Note: For dual and quad OP-Amps, the typical supply current is specified in the datasheet per amplifier. What's measured in this circuit is the sum of the supply currents for all OP-Amps together.
Rail-to-Rail behaviour	Switch on the 10kHz generator. Close the switch at the 1MΩ resistor. Show input and output voltage on the oscilloscope. Modify the input voltage and compare with the output voltage.
Input offset voltage	Switch off the 10kHz generator. Set the input voltage in the middle of the range. Close the switch at the 1MΩ resistor. Connect a voltmeter between input and output. That's the offset voltage. The offset voltage may be different for different input voltage.
Input bias current	Same as above, but open the switch at the 1MΩ resistor. If the offset voltage changes, then the input current isn't zero. $I_{INPUT} = \Delta U / 1M\Omega$ With a 3 1/2 digit multimeter in 200mV range the voltage resolution is 0.1 mV and the current resolution becomes 0.1 nA. When testing a dual or quad OP-Amp, the measured current is the sum of the bias currents from 2 or 4 inputs.
Slew rate	Switch on the 10kHz generator. Close the switch at the 1MΩ resistor. Measure the slew rate of the output signal with an oscilloscope.



The bias current is the current which flows into each input, whereas the offset current is the difference between the two bias currents.

Input offset current is also explained here:
https://developerhelp.microchip.com/xwiki/bin/view/products/amplifiers-linear/operational-amplifier-ics/introduction/input-bias-offset-current/

How to decide if an OP-Amp is genuine or fake, all data at 25°C:

Op-Amp	Supply Voltage	Supply Current			Input Offset Voltage		Input Bias Current		Slew Rate		Data sheet
		typ.	max.	@	typ.	max.	typ.	max.	min.	typ.	
AD811	9 V - 36 V	16.5 mA	18 mA	30 V	0.5 mV	3 mV	2 µA	10 µA		400 V/µs	https://www.analog.com/media/en/technical-documentation/data-sheets/ad811.pdf
OP07C	6 V - 36 V	2.7 mA	5 mA	30 V	60 µV		1.8 nA			0.3 V/µs	https://www.ti.com/lit/ds/symlink/op07.pdf
OP07D						150 µV		12 nA			
OPA2277PA	4 V - 36 V	1.6 mA	1.65 mA	30 V	20 µV	50 µV	0.5 nA	2.8 nA		0.8 V/µs	https://www.ti.com/lit/ds/symlink/opa277.pdf
OPA4277PA		3.2 mA	3.3 mA								
LT1356CN	5 V - 36V	4 mA	5 mA	30 V	0.3 mV	0.8 mV	80 nA	300 nA	200 V/µs	400 V/µs	https://www.analog.com/media/en/technical-documentation/data-sheets/13556fc.pdf
TL081BCN	4.5 V - 40 V	1.4 mA	2.5 mA	30 V	2 mV	3 mV	65 pA	200 pA	8 V/µs	20 V/µs	https://www.ti.com/lit/ds/symlink/tl084h.pdf Almost all cheap chips on Ebay or Aliexpress are counterfeit chips with TI logo
TL082BCN		2.8 mA	5 mA								
TL084BCN		5.6 mA	10 mA								

OP-Amp chips with these date codes were tested as fake:

Text on chip	Seller
(TI Symbol) 29A53LME4 TL084BCN	Aliexpress "Chipssupply Store" https://cicotex.aliexpress.com/store/1967551
(TI Symbol) 18MDSHY TL081CP	Ebay "mecklenburg8" https://www.ebay.de/str/mecklenburg8

17.2 Power OP-Amps

Type	Supply Voltage	Offset Voltage		Output Current	Slew Rate	Package	Link to Datasheet or Demoboard, Notes
		Typical	Maximum				
OPA541	+-5 V to +-40 V	+-2 mV	+-10 mV	5 A cont, 10 A peak	10 V/µs	11-pin TO-220	https://www.ti.com/lit/ds/symlink/opa541.pdf https://de.aliexpress.com/item/1005001523037649.html https://de.aliexpress.com/item/1005005046010895.html
OPA547	+-4 V to +-30 V	+-1 mV	+-5 mV	0.5 A cont, 0.75 A peak	6 V/µs	7-pin TO-220	https://www.ti.com/lit/ds/symlink/opa547.pdf https://www.mouser.de/ProductDetail/Texas-Instruments/OPA547T?qs=wgAEGBTxy7lXRdshxoRiSg%3D%3D
OPA548	+-4 V to +-30 V	+-2 mV	+-10 mV	3 A cont, 5 A peak	10 V/µs	7-pin TO-220	https://www.ti.com/lit/ds/symlink/opa548.pdf https://de.aliexpress.com/item/1005005046010895.html
OPA549	+-4 V to +-30 V	+-1 mV	+-5 mV	8 A cont, 10 A peak	9 V/µs	11-pin power	https://www.ti.com/lit/ds/symlink/opa549.pdf https://de.aliexpress.com/item/1005005046010895.html
OPA2544T	+-10 V to +-35 V	+-1 mV	+-5 mV	2 A	8 V/µs	11-pin power	https://www.ti.com/lit/ds/symlink/opa2544.pdf

17.3 Difference Amplifiers

Type	Supply Voltage	Offset Voltage		Input Common Mode Range at full supply voltage	Differential Gain [V/V]	Package	Link to Datasheet or Demoboard, Notes
		Typical	Maximum				
AD628	4.5 V to 36 V		+-1.5 mV	+-120 V	0.1 to 100	SO8	https://www.analog.com/media/en/technical-documentation/data-sheets/AD628.pdf
INA143	4.5 V to 36 V	+-100µV	+-250µV	+-18.8 V	0,1 or 10	SO8	https://www.ti.com/lit/ds/symlink/ina143.pdf End-of-Life Product
INA146	4.5 V to 36 V	+-1 mV	+-5 mV	+-100 V	0.1 to 100	SO8	https://www.ti.com/lit/ds/symlink/ina146.pdf

17.4 Microcontroller boards

	ADCs	DACs	RAM	Flash ROM	CPU Clock	Microcontroller	Notes and Links
Teensy 2.0	12 x 10-bit	-	2.5 kB	31.5 kB	16 MHz	ATMEGA32U4 AVR	no longer available https://www.pjrc.com/store/teensy.html
Teensy LC	13 x 16-bit	1 x 12-bit	8 kB	62 kB	48 MHz	MKL26Z64VFT4 Cortex-M0+	no longer available https://www.pjrc.com/store/teensylc.html
Teensy 4.0	14 x 12-bit	-	1 MB	1984 kB	600 MHz	IMXRT1062DVL6 Cortex-M7	Floating point math unit, 64 & 32 bits https://www.pjrc.com/store/teensy40.html
Teensy 4.1	18 x 12-bit	-	1 MB + 2·8 MB	7936 kB	600 MHz	IMXRT1062DVJ6 Cortex-M7	Floating point math unit, 64 & 32 bits https://www.pjrc.com/store/teensy41.html
Arduino Nano ESP32	8 x	-	512 kB	384 kB	up to 240MHz	ESP32-S3	https://store.arduino.cc/collections/boards-modules/products/nano-esp32
Arduino Nano 33 IoT	8 x 12-bit	1 x 10-bit	32 kB	256 kB	48 MHz	SAMD21 Cortex®-M0+	https://store.arduino.cc/products/arduino-nano-33-iot
Arduino UNO R4 Minima	6 x 14-bit	1 x 12-bit	32 kB	256 kB	48 MHz	Renesas RA4M1	Has a single-precision FPU https://store.arduino.cc/collections/boards-modules/products/uno-r4-minima
Arduino Due	12 x 12-bit	2 x 12-bit	96 kB	512 kB	84 MHz	Atmel SAM3X8E ARM Cortex-M3 CPU	https://store.arduino.cc/collections/boards-modules/products/arduino-due-without-headers DAC example: https://docs.arduino.cc/tutorials/due/simple-waveform-generator/
Arduino Giga R1 Wifi	12 inputs, 3 16-bit ADCs	2 x 12-bit	1 MB	2 MB	480 MHz and 240 MHz	Dual core Cortex®-M7 Cortex®-M4	https://store.arduino.cc/products/giga-r1-wifi 800x480 Display: https://docs.arduino.cc/resources/datasheets/ASX00039-datasheet.pdf
Arduino Portenta C33	7 x 12-bit	2 x 12-bit	512 kB	2 MB	200 MHz	ARM Cortex-M33 CPU	https://docs.arduino.cc/hardware/portenta-c33/
Portenta H7 Lite	11 inputs, 3 16-bit ADCs	2 x 12-bit but only one has a pin?	1 MB	2 MB	480 MHz and 240 MHz	STM32H747 dual core Cortex® M7 Cortex® M4	https://store.arduino.cc/products/portenta-h7-lite https://docs.arduino.cc/resources/datasheets/ABX00042-ABX00045-ABX00046-datasheet.pdf

	ADCs	DACs	RAM	Flash ROM	CPU Clock	Microcontroller	Notes and Links
Seeed XIAO SAMD21	11 x 12-bit	1 x 10-bit	32 kB	256 kB	48 MHz	ATSAMD21G18A-MU	https://wiki.seeedstudio.com/Seeeduino-XIAO/
Seeed XIAO RA4M1	14 x 14-bit	1 x 12-bit	32 kB	256 kB	48 MHz	ARM® Cortex®-M4 R7FA4M1AB3CFM	Has a single-precision FPU https://wiki.seeedstudio.com/getting_started_xiao_ra4m1/ RA4M1 Datasheet: https://www.renesas.com/us/en/document/dst/ra4m1-group-datasheet RA4M1 Users manual: https://www.renesas.com/us/en/document/mah/renesas-ra4m1-group-users-manual-hardware?r=1054146
Weact RA4M1	25 x 14-bit	1 x 12-bit	32 kB	256 kB	48 MHz	ARM® Cortex®-M4 R7FA4M1AB3CFM	It's unclear which IDE can be used https://de.aliexpress.com/item/1005006103872563.html https://de.aliexpress.com/item/1005007731802667.html
Spotpear REA4M1 ZERO							Instructions for Arduini IDE: https://www.waveshare.com/wiki/RA4M1-Zero Install "UNO R4" in Arduino IDE and select "UNO R4 Minima" https://de.aliexpress.com/item/1005008973289521.html https://spotpear.com/shop/RA4M1-Tiny-SuperMini-Arduino-Uno-R4-Minima-ZERO.html
Spotpear REA4M1 TINY							Instructions for Arduini IDE: https://www.waveshare.com/wiki/RA4M1-Zero Install "UNO R4" in Arduino IDE and select "UNO R4 Minima" https://de.aliexpress.com/item/1005008419941198.html https://spotpear.com/shop/RA4M1-Tiny-SuperMini-Arduino-Uno-R4-Minima-R7FA4M1.html
Arduino Zero SAMD21	6 x 12-bit	1 x 10-bit	32 kb	256 kB	48 MHz	ATSAMD21G18, 32-Bit ARM® Cortex® M0+	https://store.arduino.cc/products/arduino-zero
GY SAMD21	14 x 12-bit	yes	32 kB	256 kB	48 MHz	ATSAMD21G18	https://de.aliexpress.com/i/32975308152.html I found no documentation at all!
Sparkfun SAMD21	14 x 12-bit	1 x 10-bit	32 kB	256 kB	48 MHz	ATSAMD21G18	https://www.sparkfun.com/products/13664 Some documentation: https://learn.sparkfun.com/tutorials/samd21-minidev-breakout-hookup-guide

	ADCs	DACs	RAM	Flash ROM	CPU Clock	Microcontroller	Notes and Links
JOY-IT NodeMCU ESP32	15 x 12-bit	2 x 8-bit	512 kB	4 MB	240 MHz	Tensilica LX6 Dual-Core	https://joy-it.net/de/products/SBC-NodeMCU-ESP32 I didn't find much documentation for this board! DAC resolution is 8-bit as described here: https://circuits4you.com/2018/12/31/esp32-dac-example/ ADC resolution is 12-bit: https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
ESP-WROOM-32	18 x 12-bit	2 x 8-bit					https://de.aliexpress.com/item/1005006474252674.html https://www.elektor.de/amfile/file/download/file/1674/product/8641/
Adafruit QT Py ESP32 Pico	10 x 12-bit	2 x 8-bit	2 MB	8 MB	240 MHz	ESP32 V2 03 Dual Core	https://www.adafruit.com/product/5395
Elecrow 3.5"-HMI ESP32 Display 480x320 SPI TFT LCD Touch Screen	?	?	512 kB	4 MB	240 MHz	ESP32-WROVER-B	https://www.elecrow.com/esp32-display-3-5-inch-hmi-display-spi-tft-lcd-touch-screen.html
Feather ESP32-S2 with TFT-Display, Adafruit P5300	15 x 12-bit ?	2 x 8-bit	2 MB	4 MB	240 MHz	ESP32-S2	has a 240x135 color TFT display https://botland.de/wifi-und-bt-module-esp32/23369-feather-esp32-s2-mit-tft-display-wifi-modul-gpio-4mb-flash-2mb-psram-adafruit-p5300.html

Teensy tech specs: https://www.pjrc.com/teensy/techspecs.html

Arduino / Teensy SPI Library: https://www.arduino.cc/reference/en/language/functions/communication/spi/

Teensy 4.0 and 4.1 has 1MB RAM. It can be extended by 8MB or 16MB with this chip: https://www.pjrc.com/store/psram.html

17.5 ADC Chips

1-Channel ADC chips with SPI output:

Chip	Channels	Resolution	SR	Package	Price and Link	Datasheet	Notes
MCP3201	1	12-bit	100kSPS		https://www.aliexpress.com/item/1005005973975124.html	http://ww1.microchip.com/downloads/en/DeviceDoc/21290F.pdf	SPI
MCP3301	1	13-bit	100kSPS			https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21700E.pdf	SPI

4-Channel ADC chips with SPI output:

Chip	Channels	Resolution	SR	Package	Price and Link	Datasheet	Notes
ADS1118	4	16-bit	860SPS		https://de.aliexpress.com/item/1005003830314964.html	https://www.ti.com/lit/ds/symlink/ads1118.pdf	SPI, Delta-Sigma, programmable gain
MCP3204	4	12-bit	100kSPS			https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21298e.pdf	SPI, Successive Approximation
MCP3302	4 (unipolar) 2 (bipolar)	12-bit (unipolar) 13-bit (bipolar)	100kSPS	DIP14, SO14	Reichelt 4.92 EUR https://www.reichelt.de/de/de/shop/produkt/13-bit_serieller_a_d-wandler_2-kanal_diff_-eingang_so-14-280405	https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21697F.pdf	SPI, Successive Approximation

4-Channel ADC chips with I2C output:

Chip	Channels	Resolution	SR	Package	Price and Link	Datasheet	Notes
ADS1115	4	16-bit	860SPS		https://www.aliexpress.com/item/1005005973975124.html	https://www.ti.com/lit/ds/symlink/ads1115	I2C, Delta-Sigma, programmable gain

8-Channel ADC chips with SPI output:

Chip	Channels	Resolution	SR	Package	Price and Link	Datasheet	Notes
AD7606B STZ	8	16-bit	200kSPS	LQFP 64 0.5mm	38.43 EUR https://de.farnell.com/analog-devices/ad7606bstz/a-d-wandler-16-bit-200ksps-lqfp/dp/4016556?st=ad7606 Evaluation Board: https://de.farnell.com/analog-devices/eval-ad7606bfmcz/evaluationsboard-adc/dp/4032769?st=ad7606 Very cheap board from Aliexpress: https://de.aliexpress.com/item/4001307685007.html https://de.aliexpress.com/item/1005006303666676.html	https://www.analog.com/media/en/technical-documentation/data-sheets/ad7606_7606-6_7606-4.pdf Evaluation Board: https://www.analog.com/media/en/technical-documentation/user-guides/eval-ad7606bfmcz-ug-1225.pdf Problem: There is no SPI library for Teensy with more than 1 data line. Either use SPI with one data line (reduced throughput) or 16-bit parallel interface.	simultaneous Sampling, +-10V Bipolar Input
AD7607	8	14-bit	200kSPS	LQFP 64 0.5mm	32.68 EUR https://de.farnell.com/analog-devices/ad7607bstz/a-d-wandler-14-bit-200ksps-lqfp/dp/4016565?ost=ad7607	https://www.analog.com/media/en/technical-documentation/data-sheets/ad7607.pdf	Simultaneous Sampling, +-10V Bipolar Input
AD7761	8	16-bit	256kSPS	LQFP 64 0.5mm ???	20.48 EUR https://de.farnell.com/analog-devices/ad7761bstz/a-d-wandler-16bit-256ksps-lqfp/dp/4016745?ost=ad7761	https://www.analog.com/media/en/technical-documentation/data-sheets/AD7761.pdf	Simultaneous Sampling, Sigma-Delta, complicated

Chip	Channels	Resolution	SR	Package	Price and Link	Datasheet	Notes
ADAS3023	8	16-bit	125kSPS	LFCSP40 0.5mm	36.88 EUR https://de.farnell.com/analog-devices/adas3023bcpz/a-d-wandler-16bit-125ksps-lfcsp/dp/4017219?ost=adas3023	https://www.analog.com/en/products/adas3023.html	Simultaneous Sampling
LTC2345-16	8	16-bit	200kSPS	?	?	https://www.analog.com/media/en/technical-documentation/data-sheets/234516f.pdf	Differential inputs
LTC2358-16	8	16-bit	200kSPS	LQFP 48 0.5mm	39.24 EUR https://de.farnell.com/analog-devices/ltc2358clx-16-pbf/a-d-wandler-16-bit-200ksps-lqfp/dp/4018751?st=ltc2358	https://www.analog.com/media/en/technical-documentation/data-sheets/ltc2358-16.pdf	Differential ±10.24V
MCP3208	8	12-bit	100kSPS			https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21298e.pdf	SPI, Successive Approximation
MCP3304	8 (unipolar) 4 (bipolar)	12-bit (unipolar) 13-bit (bipolar)	100kSPS	DIP16, SO16	Reichelt 5.63 EUR https://www.reichelt.de/de/de/shop/produkt/13-bit_serieller_a_d-wandler_4-kanal_diff_-eingang_dip-16-280408	https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21697F.pdf	SPI, Successive Approximation

Sample code for Teensy for reading AD7606 with SPI interface (one data line):

Source: https://forum.pjrc.com/index.php?threads/spi-with-2-or-4-data-lines.74015/

```
#include <SPI.h>

uint32_t startTime, elapsedTime, old_res;
#define SCALE_FACTOR 0.000152587890625
/* D14 and D15 should be grounded if you use this ad7606 board
 *  https://www.amazon.de/RELAND-SUN-Multi-Channel-Datenerfassungsmodul-Synchronisation/dp/B0B4HKSK8B/ref=sr_1_2?keywords=ad7606&qid=1681803018&sr=8-2
 *  i do not use oversampling in this example
 *  i set range to the ground in this example so the range is +/-5V
*/
#define BUSY 32               // you can use any digital pin
#define RESET 30              // you can use any digital pin
#define START_CONVERSION 34   // you can use any digital pin
#define CHIP_SELECT 10        // SPI CS     // #define CHIP_SELECT 53        // SPI CS
#define D7_out 12             // SPI MISO there is no need to use MOSI port with the ad7606 #define D7_out 50            // SPI MISO there is no need
to use MOSI port with the ad7606
#define RD 13                 // SPI SCLK #define RD 52                // SPI SCLK
#define RANGE 36               // you can use any digital pin
#define TOTAL_RAW_BYTES 16
SPISettings _spiSettings;

int bytesToRead = TOTAL_RAW_BYTES;
byte raw[TOTAL_RAW_BYTES];
uint16_t parsed[8];


//--------------SETUP-UP----------------------------
void setup()
{
    initial();
    Serial.begin(2000000);
    while (!Serial) {} // wait for usb connection
    SPI.begin();
}
//---------------LOOP--------------------------
void loop()
{
    startTime = micros();
    readData();
    elapsedTime = micros() - startTime;
    if (elapsedTime != old_res)
    {
        Serial.print("Conversion time:\t");
        Serial.print(elapsedTime);
        Serial.println(" microseconds");
        old_res = elapsedTime;
```

```
    }
    parseRawBytes();

    for (int i = 0; i < 8; i++) {
        Serial.print((float)parsed[i] * SCALE_FACTOR, 5);
        Serial.print(",");
    }

    Serial.print("\r\n");
}
//----------------------------------------
void initial()
{
    pinMode(BUSY, INPUT);
    pinMode(RESET, OUTPUT);
    pinMode(START_CONVERSION, OUTPUT);
    pinMode(CHIP_SELECT, OUTPUT);
    pinMode(D7_out, OUTPUT);
    pinMode(RD, OUTPUT);
    digitalWrite(START_CONVERSION, HIGH);
    digitalWrite(CHIP_SELECT, HIGH);
    reset(RESET);
}
//----------------------------------------
void parseRawBytes() {
    /*
    I figured the problem should be here in the function of parseRawBytes.
    The sizeof(int) is 4 in Portanta and also Giga.
    This results in the value of (sizeof(parsed) / sizeof(int)) is 4 not 8,
    so only the data for the first 4 channels can be obtained.
    I modify the maximum value of loop, eight channels data can be obtained.

        for (int i = 0; i < (sizeof(parsed) / sizeof(int)); i++)
        {
            parsed[i] = (raw[i * 2] << 8) + raw[(i * 2) + 1];
        }
    */

    for (int i = 0; i < 8; i++)
    {
        parsed[i] = (raw[i * 2] << 8) + raw[(i * 2) + 1];
    }
}
//----------------------------------------
/*reset signal*/
void reset(uint8_t port)
{
    digitalWrite(port, HIGH);
    // delayMicroseconds(1);
```

```
    digitalWrite(port, LOW);
    //  delayMicroseconds(1);
}
//----------------------------------------
void conversionPulse(uint8_t port)
{
    digitalWrite(port, LOW);
    //  delayMicroseconds(1);
    digitalWrite(port, HIGH);
}
//----------------------------------------
/*
1- start conversion START_CONVERSION HIGH
2- wait until the busy is LOW again
3- put the CS to LOW
4- Read a byte from the SPI
*/
void readData()
{
    conversionPulse(START_CONVERSION);
    while (digitalRead(BUSY) == HIGH) {
        //  delayMicroseconds(1);
    }
    SPI.beginTransaction(_spiSettings);
    digitalWrite(CHIP_SELECT, LOW);
    while (bytesToRead > 0) {
        raw[TOTAL_RAW_BYTES - bytesToRead] = SPI.transfer(0x00);
        bytesToRead--;
    }
    digitalWrite(CHIP_SELECT, HIGH);
    SPI.endTransaction();

    bytesToRead = TOTAL_RAW_BYTES;
}
//----------------------------------------
/*
  0 +/-  5V
  1 +/- 10V
*/
void setRange(bool range)
{
    pinMode(RANGE, OUTPUT);
    digitalWrite(RANGE, range);
}
//----------------------------------------
/*OS2 OS1 OS0
oversampling
000 No oversampling Maximum sampling rate is 200KSPS
001  2 times        100 KSPS
```

```
010  4 times        50 KSPS
011  8 times        25 KSPS
100 16 times        12.5 KSPS
101 32 times        6.25 KSPS
110 64 times        3.125 KSPS
*/
/*
void setOversampling(uint8_t OS0, uint8_t OS1, uint8_t OS2) {
    pinMode(_OS0, OUTPUT);
    pinMode(_OS1, OUTPUT);
    pinMode(_OS2, OUTPUT);
    digitalWrite(_OS0, bitRead(B001, OS0));
    digitalWrite(_OS1, bitRead(B010, OS1));
    digitalWrite(_OS2, bitRead(B100, OS2));
}
*/
```

GPIO6_PSR is the hardware register that must be read:

uint32_t port6_val = GPIO6_PSR;

```
//MXRT pin  |6.31|6.30|6.29|6.28|6.27|6.26|6.25|6.24|6.23|6.22|6.21|6.20|6.19|6.18|6.17|6.16|
//4.1 pins  |27  |26  |39  |38  |21  |20  |23  |22  |16  |17  |41  |40  |15  |14  |18  |19  |
```

If you can just leave the 16 pins as input mode, I would recommend first configuring all 16 with ordinary pinMode(). There are important low-level details about pin configuration which are handled automatically by pinMode().

sMore interesting infos from Pauf Stoffregen: (Source: https://forum.pjrc.com/index.php?threads/spi-with-2-or-4-data-lines.74015/)

In the reference manual you'll find a lot of documentation which says these pins are connected to GPIO1. Then elsewhere you can find info about the ability to map a pin to either the slow GPIO (1 to 5) or fast GPIO (6 to 9). But almost all the documentation was written for the earlier RT1052 chip, so a lot of documentation mentions GPIO1 only. Of course, knowing how the RT1062 hardware works, you can know that any pins GPIO1 accesses can alternately be accessed by GPIO6.

Teensy's startup code maps all the pins to the fast GPIO peripherals. So even though the default for non-Teensy boards would be GPIO1, because of the way the Teensy core library is designed, you would access the port with GPIO6 rather than GPIO1.

Not documented in the reference manual is the (likely) story behind *why* the chip has redundant GPIO peripherals. NXP's earlier RT1052 chip had only the slow GPIO. They almost certainly heard feedback from (huge corporation) customers about the slow GPIO performance. In the design of these modern chips, it's much easier to bolt on another peripheral than to dive deep down into the design of the existing ones. And the slowness comes from various buses and bus bridges, which might be IP they licensed from ARM which often comes with restrictions on modifications (which is a smart policy for ARM as it keeps a lot of consistency in the platform and required compiler support). So rather than do anything about the GPIO slowness, they just added another copy of the same GPIO perhipheral but connected it to a special low-latency bus the M7 core provides. And then they added muxes to let

you assign each pin to either the slow or fast GPIO.

The main downside to the fast GPIO is DMA can't access anything on the low latency bus (all 0x42xxxxxx addresses). So when a pin needs to accessed by DMA, as the OctoWS2811 library does, the special mux registers are used to assign those pins to the slow GPIO which DMA can access.

Other than DMA, you would normally always want to the fast GPIO. That's why Teensy's startup code assigns all the pins to be used by fast GPIO6 to GPIO9.

17.6 DAC Chips

DAC chips with SPI input:

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5328	8	12-Bit	5.5 V			https://www.mouser.de/datasheet/2/609/AD5308_5318_5328-1501620.pdf	
AD5363	8	14-Bit	+-10V		???	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5362_5363.pdf	
AD5648	8	14-Bit	0 - 2.5V or 0 - 5.0V	TSSOP 16 0.65mm	22.41 EUR https://de.farnell.com/analog-devices/ad5648aruz-2/d-a-wandler-14bit-tssop-16-40/dp/4017484?st=ad5648	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5628_5648_5668.pdf	with 5 ppm/°C, On-Chip Reference
AD5668	8	16-Bit	0 - 2.5V or 0 - 5.0V	TSSOP 16 0.65mm	26.18 EUR https://de.farnell.com/analog-devices/ad5668aruz-2/d-a-wandler-16bit-tssop-16-40/dp/4017526?st=ad5668	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5628_5648_5668.pdf Arduino Library: https://www.arduino.cc/reference/en/libraries/ad56x8/	with 5 ppm/°C, On-Chip Reference
AD5676	8	16-Bit		TSSOP 20 0.65mm	22.91 EUR https://de.farnell.com/analog-devices/ad5676bruz/d-a-wandler-16bit-tssop/dp/4017544?st=ad5676	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5676.pdf	no on-chip reference
AD5678	4+4	4×12-Bit + 4×16-Bit		TSSOP 14 0.65mm	???	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5678.pdf	On-Chip Reference

DAC8563	2	16-Bit	0 - 5.0V	VSSOP 10	https://de.aliexpress.com/item/4001192930542.html https://de.aliexpress.com/item/1005008731323360.html	https://www.ti.com/lit/ds/symlink/dac8563.pdf?ts=1708357513004	On-Chip Reference
DAC8830 DAC8831	1	16-Bit			https://de.aliexpress.com/item/1005008721374287.html https://de.aliexpress.com/item/1005008524110991.html	https://www.ti.com/lit/ds/symlink/dac8830.pdf	
DAC60508	8	12-bit				https://www.ti.com/lit/ds/symlink/dac60508.pdf	Used in Lucidac for initial conditions of the integrators
LTC2600	8	16-Bit	Rail-to-Rail	SSOP 16 0.635mm	28.18 EUR https://de.farnell.com/analog-devices/ltc2600cgn-trpbf/d-a-wandler-16bit-nssop-16/dp/4218020?st=ltc2600	https://www.analog.com/media/en/technical-documentation/data-sheets/2600fe.pdf	
LTC2656	8	16-bit or 12-bit	Rail-to-Rail		I found only the 12-bit version	https://www.analog.com/media/en/technical-documentation/data-sheets/2656fa.pdf	with 10ppm/°C Max Ref
LTC2662	5	12-bit	Current output, needs external OP-Amp	QFN32		https://www.analog.com/media/en/technical-documentation/data-sheets/ltc2662.pdf	
LTC2672	5	12-bit	Current output, needs external OP-Amp	LFCSP		https://www.analog.com/media/en/technical-documentation/data-sheets/ltc2672.pdf	
TDA1543	2	16-bit		DIL8, SO16	very cheap https://de.aliexpress.com/item/1005001682368333.html	https://www.digchip.com/datasheets/parts/datasheet/364/TDA1543-pdf.php	very old design, I2S, 5V

Multiplying DACs:

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5415	2	12-Bit	Current output, needs external OP-Amp	TSSOP24	Mouser: 10.40 EUR (25)	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5415.pdf	Daisy chaining is possible, has internal matched resistors for bipolar output
AD5449	2	12-Bit	Current output, needs external OP-Amp	TSSOP16	Mouser: 8.50 EUR (25) looks like the best choice	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5429_5439_5449.pdf	Daisy chaining is possible, needs external matched resistors for bipolar output
AD5452	1	12-bit	Current output, needs external OP-Amp	MSOP 8	Mouser: 5.44 EUR (25)	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5450_5451_5452_5453.pdf	Used in Lucidac for coefficients, one chip per lane
AD5453	1	14-bit	Current output, needs external OP-Amp	MSOP 8	https://de.aliexpress.com/item/1005004181362052.html	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5450_5451_5452_5453.pdf	Multiplying DAC, Vref=+-10V
AD5543	1	16-bit	Current output, needs external OP-Amp	MSOP8	https://de.aliexpress.com/item/1005002443988309.html	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5543_5553.pdf	Multiplying DAC, Vref=+-10V

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5544	4	16-bit	Current output, needs external OP-Amp	SSOP 28	53.49 EUR https://de.farnell.com/analog-devices/ad5544arsz-reel7/d-a-wandler-16bit-ssop-28-40-bis/dp/4023048?st=ad5544	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5544_5554.pdf	Multiplying DAC, Vref=+-10V Compatible with DAC8814
AD5554	4	14-Bit	Current output, needs external OP-Amp	SSOP 28	Mouser: 34.96 EUR (10) 38.89 EUR https://de.farnell.com/analog-devices/ad5554brsz/d-a-wandler-14bit-ssop-28-40-bis/dp/4023062?ost=ad5554	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5544_5554.pdf	Multiplying DAC Vref=+-10V Compatible with DAC8803
AD7564	4	12-bit	Current output, needs external OP-Amp	DIP28 SO28 SSOP28	Mouser: 31.11 EUR (25)	https://www.analog.com/media/en/technical-documentation/data-sheets/AD7564.pdf	
AD7568	8	12-bit	Current output, needs external OP-Amp	PQFP44 PLCC44	Mouser: 86.84 EUR (1) Mouser: 80.13 EUR (10)	https://www.analog.com/media/en/technical-documentation/data-sheets/AD7568.pdf	
DAC8803	4	14-Bit	Current output, needs external OP-Amp	SSOP 28	25.83 EUR https://www.mouser.de/ProductDetail/Texas-Instruments/DAC8803IDBT?qs=vul0MIC%2Fa1cIPELYpZUEyA%3D%3D	https://www.ti.com/lit/ds/symlink/dac8803.pdf	Multiplying DAC Vref=+-10V Compatible with AD5554
DAC8814	4	16-bit	Current output, needs external OP-Amp	SSOP 28	34.40 EUR https://www.digikey.de/de/products/detail/texas-instruments/DAC8814ICDBT/863244	https://www.ti.com/lit/ds/symlink/dac8814.pdf	Multiplying DAC, Vref=+-10V Compatible with AD5544

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
LTC1590	2	12-bit	Current output, needs external OP-Amp	DIP16 SO16		https://www.analog.com/media/en/technical-documentation/data-sheets/1590f.pdf	

DAC chips with I2C or I2S input:

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5693R	1	16-Bit	0 - 2.5 V or 0 - 5.0 V		https://de.aliexpress.com/item/1005007519469251.html https://de.aliexpress.com/item/1005008318926961.html	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5693r_5692r_5691r_5693.pdf	I2C On-Chip Reference
DAC5578 DAC6578 DAC7578	8	8-bit 10-bit 12-bit		TSSOP16	https://www.adafruit.com/product/6258	https://www.ti.com/lit/ds/symlink/dac6578.pdf	I2C
MCP 4725	1	12-bit	Rail-to-Rail	SOT 23-6	very cheap https://de.aliexpress.com/item/1005005910501182.html https://de.aliexpress.com/item/1005005970420972.html https://de.aliexpress.com/item/4000313262947.html	https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP4725-12-Bit-Digital-to-Analog-Converter-with-EEPROM-Memory-DS20002039.pdf	I2C 2.7V-5.5V
MCP 4728	4	12-bit	Rail-to-Rail	MSOP 10	very cheap https://de.aliexpress.com/item/1005006038589972.html https://de.aliexpress.com/item/1005005797422191.html https://de.aliexpress.com/item/1005007181220423.html On the back side of the module is a normal-open jumper, which can connect $\overline{\text{LDAC}}$ to GND	https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/22187E.pdf Sample code files: https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/CodeExamples/Code_Examples_DAC.zip	I2C 2.7V-5.5V
TDA1543	2	16-bit		DIL8, SO16	very cheap https://de.aliexpress.com/item/1005001682368333.html	https://www.digchip.com/datasheets/parts/datasheet/364/TDA1543-pdf.php	I2S, very old design for audio, 5V

17.7 A/D and D/A combinations

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5592R	8	12-bit	5V	TSSOP16 0.65mm	9.12 EUR https://www.mouser.de/ProductDetail/Analog-Devices/AD5592RBRUZ?qs=wFnXgzJ2EGM0E%2FJNKZBaug%3D%3D 9.16 EUR https://de.farnell.com/analog-devices/ad5592rbruz/a-d-wandler-12-bit-400ksps-tssop/dp/4016360?st=ad5592r	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5592r.pdf Library for Teensy: https://github.com/dzalf/AD5592R	SPI interface, contains a total of 8 ADCs or DACs
AD5593R	8	12-Bit	5V	TSSOP16 0.65mm	https://de.aliexpress.com/item/1005002433701414.html 9.19 EUR https://de.farnell.com/analog-devices/ad5593rbruz-rl7/konfigurierbarer-adc-dac-12bit/dp/4016365?st=ad5593r	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5593r.pdf	I2C interface, contains a total of 8 ADCs or DACs

17.8 Digital potentiometers

Chip	Positions	R	Package	Price and Link	Datasheet
AD5293	1024 (10-bit)	20k, 50k or 100k	TSSOP 14	8.12 EUR 10 pieces: 7.32 EUR https://de.farnell.com/analog-devices/ad5293bruz-50/digitalpotentiometer-1fach-tssop/dp/4028695?st=ad5293	SPI interface https://www.analog.com/media/en/technical-documentation/data-sheets/AD5293.pdf
AD8400 AD8402 AD8403	256 (8-bit)	1k, 10k, 50k, 100k			SPI interface, used in Lucidac? https://www.analog.com/media/en/technical-documentation/data-sheets/ad8400_8402_8403.pdf
TPL0501	256 (8-bit)	100k		cheap https://de.aliexpress.com/item/1005007223925581.html	SPI interface https://www.ti.com/product/de-de/TPL0501-100

17.9 Analog switches

Chip	Switches	Supply	R_{ON}	Charge injection	Price and Link	Datasheet
ADG408	single 8:1	+-22 V	100 Ω			https://www.analog.com/media/en/technical-documentation/data-sheets/ADG408_409.pdf
ADG409	dual 4:1					
ADG417	single SPST	+-22 V	35 Ω	< 7 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/ADG417.pdf
ADG419	single SPDT	+-22 V	35 Ω	< 7 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/ADG419.pdf
ADG508	single 8:1	+-22 V	280 Ω	< 4 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/ADG508A_509A.pdf
ADG509	dual 4:1					
ADG1201	single SPST	+-17 V	120 Ω	< 0.8 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/ADG1201.pdf
ADG1206	single 16:1	+-17 V	120 Ω	< 1 pC	1.99 EUR (several batches tested ok) https://de.aliexpress.com/item/1005004974570872.html	https://www.analog.com/media/en/technical-documentation/data-sheets/ADG1206_1207.pdf
ADG1207	dual 8:1				56.08 EUR / 10pcs https://de.aliexpress.com/item/1005007047020576.html	
ADG5208	single 8:1	+-22 V	250 Ω	< 0.4 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/adg5208f_5209f.pdf
ADG5209	dual 4:1					
CD4066B	quad SPST	+-9 V	125 Ω		cheap, but not suitable for +-10V range	https://www.ti.com/lit/ds/symlink/cd4066b.pdf

DG212B (used in THAT)	Quad SPST normal open	+-18 V	Vishay: 50 Ω Analog Devices: 115 Ω		0.50 EUR (tested ok) https://de.aliexpress.com/item/1005003762068601.html 1.03 EUR (tested ok) https://de.aliexpress.com/item/1005007286059104.html 0.59 EUR (tested ok) https://de.aliexpress.com/item/1005007620742208.html	https://www.vishay.com/docs/61556/dg211.pdf https://www.analog.com/media/en/technical-documentation/data-sheets/DG202-DG212.pdf
DG213	quad SPST 2x normal open 2x normal closed	+-22 V	45 Ω	< 1 pC	Mouser 2.51 EUR	https://www.vishay.com/docs/70662/dg213.pdf
DG408	single 8:1	+-22 V	100 Ω	< 15 pC	3.90 EUR https://www.reichelt.de/single-8-kanal-cmos-analog-multiplexer-dil-16-dg-408-djz-p188815.html	https://www.analog.com/media/en/technical-documentation/data-sheets/DG408-DG409.pdf
DG409	dual 4:1				2.95 EUR https://www.reichelt.de/differenzielle-4-kanal-cmos-analog-multiplexer-dil-16-dg-409-djz-p188817.html	
DG413	quad SPST	+-20 V	35 Ω	< 5 pC	5.55 EUR https://www.reichelt.de/analog-schalter-ic-4-kanal-dil-16-dg-413-djz-p188821.html	https://cdn-reichelt.de/documents/datenblatt/A200/DG411_412_413-ITS.pdf
DG417	single SPST normal closed	+-20 V	20 Ω	< 3 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/DG417-DG419.pdf https://www.analog.com/media/en/technical-documentation/data-sheets/ADG417.pdf
DG418	single SPST normal open					

DG419	single SPDT				Reichelt 5.90 EUR https://www.reichelt.de/index.html?ACTION=446&LA=446&nbc=1&q=dg419 5.49 EUR / 10 pieces https://de.aliexpress.com/item/1005003214167615.html	
DG441	quad SPST	+-20 V	50 Ω	< 5 pC	2.75 EUR https://www.reichelt.de/analog-schalter-ic-4-kanal-so-16-dg-441-dyz-p188824.html	https://www.analog.com/media/en/technical-documentation/data-sheets/dg441-dg442.pdf
DG442	quad SPST					
DG1206	single 16:1	+-20 V	100 Ω	< 0.5 pC	6.64 EUR https://www.mouser.de/c/?marcom=192816146	https://www.analog.com/media/en/technical-documentation/data-sheets/DG1206-DG1207.pdf
DG1207	dual 8:1					
DG1208	single 8:1	+-20 V	100 Ω	< 0.5 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/DG1208-DG1209.pdf
DG1209	dual 4:1					
MAX4511	Quad SPST normal closed	+-18 V	125 Ω	1.5 pC		https://www.analog.com/media/en/technical-documentation/data-sheets/MAX4511-MAX4513.pdf
MAX4512	Quad SPST normal open					
MAX4513	quad SPST 2x normal open 2x normal closed					

17.10 Analog Crosspoint Switches

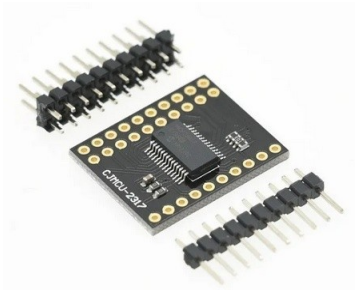
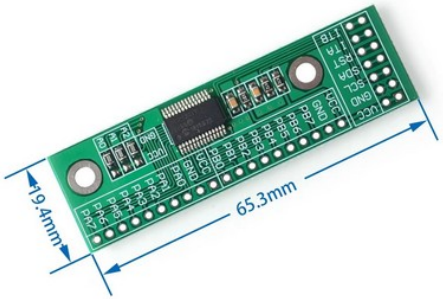
Chip	Switches	Supply	R_{ON}	Price and Link	Datasheet
AD75019	16x16	+-12 V	150 Ω	10.39 EUR tested ok https://de.aliexpress.com/item/1005003236251200.html	https://www.analog.com/media/en/technical-documentation/data-sheets/AD75019.pdf
MAX4358	32x16	+-5 V	?		https://www.analog.com/media/en/technical-documentation/data-sheets/MAX4358.pdf
MT8816	8 x 16	4.5 V - 13.2 V	65 Ω	< 10 EUR https://www.mouser.de/c/semiconductors/integrated-circuits-ics/communication-networking-ics/analog-digital-crosspoint-ics/?configuration=8 x 16&product=Analog Crosspoint Switches	https://www.mouser.de/datasheet/2/268/Microchip_06182024_MT8816AE1-3459976.pdf
CD22M3494	16 x 8	4 V - 15 V	65 Ω	https://www.mouser.de/c/semiconductors/integrated-circuits-ics/communication-networking-ics/analog-digital-crosspoint-ics/?configuration=16 x 8&product=Analog Crosspoint Switches 9.29 EUR for 3 pieces: https://de.aliexpress.com/item/32856068713.html	https://www.mouser.de/datasheet/2/698/REN_cd22m3494_DST_20000421-1997534.pdf
CH446Q	8 x 16	12 V or -7 V to +5 V	65 Ω	very cheap on Aliexpress from many sellers https://www.wch-ic.com/products/CH446.html Used in this project: https://www.hackster.io/news/prototyping-magic-hands-on-with-the-wire-free-jumperless-breadboard-8e71040c4b65	
CH446X	5 x 24				

17.11 Displays

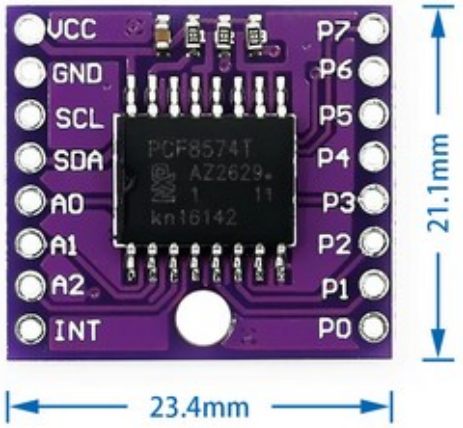
Display	Resolution	Color	Technology	Size	Chip	Links
PJRC	320x240	Color	TFT Touchscreen	2.8"	ILI9341	https://www.pjrc.com/store/display_ili9341_touch.html
Expansion Board Base for XIAO, with micro SD card slot	128x64	White	OLED	0.96"		https://wiki.seeedstudio.com/Seeeduino-XIAO-Expansion-Board/ Schematic diagram: https://files.seeedstudio.com/wiki/Seeeduino-XIAO-Expansion-Board/document/Seeeduino%20XIAO%20Expansion%20board_v1.0_SCH_200824.pdf
Elecrow 3.5"-HMI ESP32 Display	480x320	Color	TFT Touchscreen	3.5"	ILI9488	https://www.elecrow.com/esp32-display-3-5-inch-hmi-display-spi-tft-lcd-touch-screen.html
Ardi Display - IPS 2" fot Arduino Uno	240x320	Color	TFT	2"	ST7789V2	https://botland.de/arduino-shield-tastaturen-und-displays/23660-ardi-display-ips-2-240x320-px-display-overlay-fur-arduino-uno-sb-components-sku27217-5055652927217.html
Feather ESP32-S2 Adafruit P5300	240x135	Color	TFT			https://botland.de/wifi-und-bt-module-esp32/23369-feather-esp32-s2-mit-tft-display-wifi-modul-gpio-4mb-flash-2mb-psram-adafruit-p5300.html
Aliexpress	160x80	Color	TFT	0.96"	ST7735	https://de.aliexpress.com/item/1005006532468602.html
Aliexpress	240x240	Color	TFT	1.3"	ST7789	https://de.aliexpress.com/item/1005006532468602.html
Aliexpress	20x4 character	misc. oolors	LCD	77mm x 25.2mm	AIP31066, HD44780, KS0066, SPLC780D	https://de.aliexpress.com/item/1005006360823398.html I2C Interface
Aliexpress	4x40 character	misc. colors	LCD	76mm x 26mm	SPLC780	https://de.aliexpress.com/item/32991449983.html https://de.aliexpress.com/item/32908727195.html I2C Interface
Aliexpress	8 Digits 7-segment	misc. colors	LED	0.36"	74HC595	https://de.aliexpress.com/item/1005001426051500.html Multiplexing must be done by microcontroller

Aliexpress	**8 Digits 7-segment**	**misc. colors**	**LED**	**0.56"**	**74HC595**	**https://de.aliexpress.com/item/1005001425536213.html** **Multiplexing must be done by microcontroller**
Aliexpress	**6 Digits 7-segment**	**misc. colors**	**LED**	**0.36"**	**TM1637**	**https://de.aliexpress.com/item/1005004782786960.html** **TM1637 Datasheet: https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/docs/datasheet/unit/digi_clock/TM1637.pdf** **Library: https://github.com/jasonacox/TM1637TinyDisplay/blob/master/README.md**
Aliexpress	**6 Digits 7-segment**	**misc. colors**	**LED**	**0.56"**	**TM1637**	**https://de.aliexpress.com/item/1005004646813267.html** **TM1637 Datasheet: https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/docs/datasheet/unit/digi_clock/TM1637.pdf** **Library: https://github.com/jasonacox/TM1637TinyDisplay/blob/master/README.md**

17.12 I/O Expanders

	MCP23009 https://www.reichelt.de/de/de/shop/produkt/i_c-bus_i_o_erweiterung_8bit_pdip-18-266218
	MCP23016 https://ww1.microchip.com/downloads/en/DeviceDoc/20090C.pdf
 	**Bidirectional 16-bit serial / parallel I2C SPI interface MCP23017** https://de.aliexpress.com/item/1005006386066328.html https://de.aliexpress.com/item/32883688022.html **MCP23017 has I2C interface, also available in DIP28 package** **MCP23S17 has SPI interface, also available in DIP28 package (but quite expensive)** **Datasheet:** https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP23017-Data-Sheet-DS20001952.pdf **Has 14 I/O pins and two output-only pins** **The I2C version has a configurable I2C adress from 0x20 to 0x27.**

	**IO Expander for XIAO with MCP23017 chip (not available at the moment):** **https://wiki.seeedstudio.com/io_expander_for_xiao/** **The default address is 0x21. If you solder the "J2" pad, the address can be changed to 0x20.**
	MCP23018 **https://www.reichelt.de/de/de/shop/produkt/i_c-bus_i_o_erweiterung_16bit_pdip-28-266221**



PCF8574

https://www.reichelt.de/remote-8-bit-i-o-expander-for-i2c-bus-pdip-16-pcf-8574-n-p216402.html

https://de.aliexpress.com/item/1005003217797506.html

https://de.aliexpress.com/item/1005006127825537.html

https://de.aliexpress.com/item/1005003475068095.html

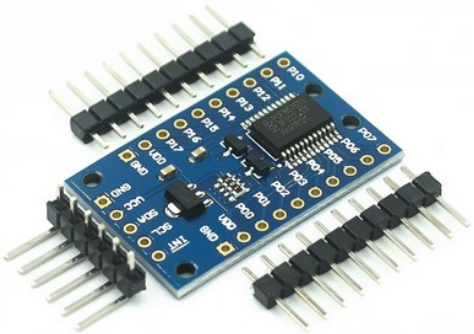
https://de.aliexpress.com/item/1005007556857939.html

Data sheet TI: https://www.ti.com/lit/ds/symlink/pcf8574.pdf

Data sheet NXP: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/847/PCF8574(A).pdf

The I2C adresses are different for the "A" version:

A2	A1	A0	PCF8574			PCF8574A		
			Write	Read	7-bit without R/$\overline{W}$	Write	Read	7-bit without R/$\overline{W}$
0	0	0	0x40	0x41	0x20	0x70	0x71	0x38
0	0	1	0x42	0x43	0x21	0x72	0x73	0x39
0	1	0	0x44	0x45	0x22	0x74	0x75	0x3A
0	1	1	0x46	0x47	0x23	0x76	0x77	0x3B
1	0	0	0x48	0x49	0x24	0x78	0x79	0x3C
1	0	1	0x4A	0x4B	0x25	0x7A	0x7B	0x3D
1	1	0	0x4C	0x4D	0x26	0x7C	0x7D	0x3E
1	1	1	0x4E	0x4F	0x27	0x7E	0x7F	0x3F

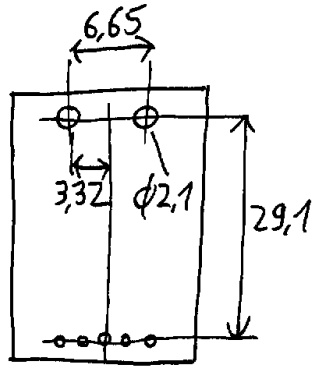
<table>
<tr>
<td>  </td>
<td>
PCF8575

https://de.aliexpress.com/item/1005006221525230.html
https://de.aliexpress.com/item/4001321236558.html
https://de.aliexpress.com/item/1005007080651955.html
https://www.reichelt.de/de/de/remote-16-bit-i-o-erweiterung-fuer-i2c-bus-ssop-24-pcf-8575-t-p70030.html

Datasheet: https://cdn-reichelt.de/documents/datenblatt/A200/PCF8575_NXP.pdf

The I2C address is configurable from 0x20 to 0x27.

Has 16 I/O pins with internal pullup. Suitable for reading 4 BCD switches without any other external components.
Onboard 3.3V level converter circuit, if you don’t have a solder pad of VCC-VDD, the PCF8575 level is 3.3V. If you connect VCC and VDD, the level will be the same as VCC.



The vertical and horizontal pin rows aren't on the same 2.54mm grid. The 2.1mm bores are also off-grid.
</td>
</tr>
<tr>
<td>  </td>
<td>
XL9535 I2C 1-2-4-8-16 channel relay boards:
https://de.aliexpress.com/item/1005005775051224.html
https://de.aliexpress.com/item/1005006147799636.html

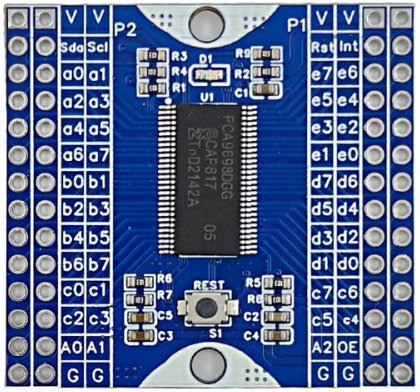
The supply voltage of the XL9535 chip is 2.3 V to 5.5 V, this voltage is on the 4-pin connectors. All three 4-pin connectors are wired parallel.

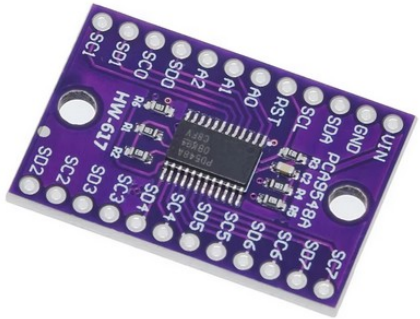
The supply voltage for the relays is 5 V, this voltage is on the hollow-plug connector and on the 2-pin screw connector and on the USB-C connector.

Both voltages can be coupled with a jumper. However it's unclear why the boards contain an optocoupler, because the XL9535 chip and the relays share the same ground level.

Datasheet of XL9535 chip:
https://www.lcsc.com/datasheet/lcsc_datasheet_2211110930_XINLUDA-XL9535_C561273.pdf

The I2C adress is configurable from 0x20 to 0x27.
</td>
</tr>
</table>

	**PCA9555** https://de.aliexpress.com/item/1005005243709565.html https://de.aliexpress.com/item/1005006889452359.html Datasheet: https://www.ti.com/lit/ds/symlink/pca9555.pdf The I2C adress is configurable from 0x20 to 0x27
	PCA9557 Datasheet: https://www.ti.com/lit/ds/symlink/pca9557.pdf The I2C adress is configurable from 0x18 to 0x1F
	**PCA9698 40 channel I/O I2C** https://de.aliexpress.com/item/1005005243724794.html https://de.aliexpress.com/item/1005006889452359.html Datasheet: https://www.nxp.com/docs/en/data-sheet/PCA9698.pdf Pins configured as inputs don't have internal pullup or pulldown resistors. The I2C adress is configurable from 0x20 to 0x5E, or from 0xA0 to 0xEE, in steps of 2

	**TCA 9548A I2C Multiplexer** https://de.aliexpress.com/item/1005004996004241.html https://de.aliexpress.com/item/1005006415966331.html https://de.aliexpress.com/item/4000495109843.html **Datasheet:** https://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=1728836842883 **The I2C adress is configurable from 0x70 to 0x77.**
	**PCA9685 16-Channel I2C servo driver** https://de.aliexpress.com/item/1005007342445289.html

17.13 Logic Level Converters

Chip	From ... to ...	Notes
74AHC125 74AHC1G125	5 V to 3.3 V	74AHC logic has 5V tolerant inputs: https://assets.nexperia.com/documents/data-sheet/74AHC_AHCT125.pdf
MOS-FET BSN20	I2C bus, bidirectional 5 V / 3.3 V	Datasheet: https://cdn-reichelt.de/documents/datenblatt/A200/BSN20BK_ENG_TDS.pdf https://de.aliexpress.com/item/1005008276554353.html Application note: https://www.eeweb.com/level-shifting-techniques-in-i2c-bus-design/ See also here: https://cdn-shop.adafruit.com/datasheets/an97055.pdf Boards are available for 2, 4 or 8 channels: https://de.aliexpress.com/item/1005005649455526.html https://de.aliexpress.com/item/1005006873109528.html https://de.aliexpress.com/item/32955186155.html
TXS0108E	bidirectional 5 V / 3.3 V	https://www.ti.com/lit/ds/symlink/txs0108e.pdf?ts=1735189112808 https://de.aliexpress.com/item/1005001619337949.html

17.14 Voltage Doublers

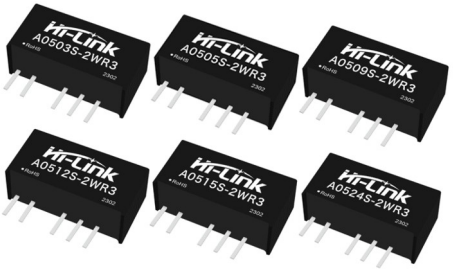
Chip	Input Voltage	Notes
MAX1682 MAX1683	2 V to 5.5 V	Datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX1682-MAX1683.pdf

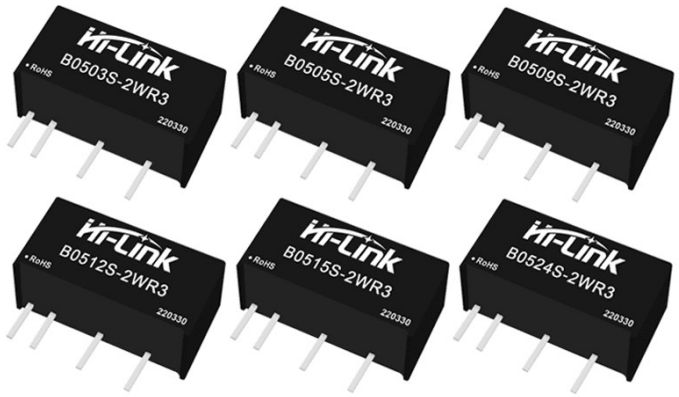
17.15 Voltage References

Chip	Voltages	Tolerance	Notes
ISL21010	1.024V, 1.25V, 1.5V, 2.048V, 2.5V, 3.0V, 3.3V, 4.096V	0.2%	https://www.reichelt.de/de/de/shop/produkt/spannungsreferenz_2_048v_-0_2_sot-23-3_t_r-386888 Datasheet: https://www.mouser.de/datasheet/2/698/REN_isl21010_DST_20210920-1998014.pdf
LM317L	adjustable 1.25 V to 32 V		Datasheet: https://www.ti.com/lit/ds/symlink/lm317l.pdf
LM336-2.5	adjustable 2.39 V to 2.59 V		Datasheet: https://www.ti.com/lit/ds/symlink/lm136-2.5-n.pdf
LM337L	adjustable negative voltage -1.2 V to -37 V		Datasheet: https://www.ti.com/lit/ds/symlink/lm337l.pdf
LM336-5.0	adjustable 4 V to 6 V		Datasheet: https://www.ti.com/lit/ds/symlink/lm336-5.0.pdf
LM385-1.2	1.235 V	2%, B: 1%	Datasheet: https://www.ti.com/lit/ds/symlink/lm385-1.2.pdf
LM385-2.5	2.5 V	3%, B: 1.5%	Datasheet: https://www.ti.com/lit/ds/symlink/lm385-2.5.pdf
LM385-ADJ = LM385Z	adjustable 1.24 V to 5.3 V		Datasheet: https://www.ti.com/lit/ds/symlink/lm385-adj.pdf
LM4040	2.048V, 2.5V, 3V, 4.096V, 5V, 8.192V, 10V	A: 0.1% B: 0.2% C: 0.5% D: 1.0%	Datasheet: https://www.ti.com/lit/ds/symlink/lm4040.pdf There are small modules available on Aliexpress with 2.048 V and 4.096 V, I'm not sure it they are really 0.1% as specified. https://de.aliexpress.com/item/1005008161362902.html
LM4041-1.2 LM4041-ADJ	1.225 V fixed and adjustable	A: 0.1% B: 0.2% C: 0.5% D: 1.0%	Datasheet: https://www.ti.com/lit/ds/symlink/lm4041c.pdf

LM4132	1.8 V, 2.048 V, 2.5 V, 3 V, 3.3 V, 4.096 V	A: 0.05% B: 0.1% C: 0.2% D: 0.4% E: 0.5%	Datasheet: https://www.ti.com/lit/ds/symlink/lm4132-q1.pdf
LT1021	5V, 7V, 10V	0.05%	Datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/1021fc.pdf
MCP1501	1.024V, 1.25V, 1.8V, 2.048V, 2.5V, 3V, 3.3V, 4.096V, 4.5V, 5V	0.1%	Datasheet: https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP1501-Data-Sheet-DS20005474.pdf
TL431 TL432	adjustable 2.5 V to 36 V		Datasheet: https://www.ti.com/lit/ds/symlink/tl431.pdf The only difference between TL431 and TL432 is different pinout.

17.16 DC-DC Converters

	**Traco TBA2-0522 (this module is used in THAT)** **https://www.tracopower.com/int/model/tba-2-0522** **https://www.reichelt.de/dc-dc-wandler-tba-2-2-w-12-v-80-ma-sil-7-tba-2-0522-p265772.html** **https://cdn-reichelt.de/documents/datenblatt/C700/TBA2-0522_DB_EN.pdf** **Input 4.5 V - 5.5 V** **Output unstabilized +-12 V 80 mA** **Pin 1: +5V** **Pin 2: GND** **Pin 3: -12 V** **Pin 4: isolated GND** **Pin 5: +12 V**
A0505S/03/09/12/15/24S-2WR3 5V to 3.3V/5V/9V/12V/15V/24V 	**Hi-Link A0512S-2WR3 or A0515S-2WR3** **https://www.hlktech.net/index.php?id=1025** **https://de.aliexpress.com/item/1005006148598010.html** **https://drive.google.com/drive/folders/1vjf-77LCVQl16Hcb1phADiU3Hz14mhYB** **Input 4.5 V - 5.5 V** **Output unstabilized +-12 V 84 mA or +-15 V 68 mA** **It's recommended to use a 4.7µH inductance at the input and at the output.** **Pin 1: +5V** **Pin 2: GND** **Pin 3: -12 V** **Pin 4: isolated GND** **Pin 5: +12 V** **Compatible modules:** **https://de.aliexpress.com/item/1005003666846199.html**

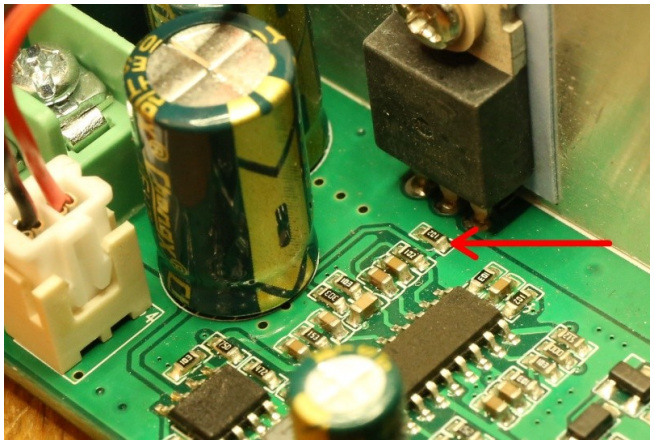
 T	**Hi-Link B0512S-2WR3 or B0515S-2WR3** **https://www.hlktech.net/index.php?id=1013** **https://de.aliexpress.com/item/1005007253701160.html** **Input 4.5 V - 5.5 V** **Output unstabilized 12 V 167 mA or 15 V 133 mA** **It's recommended to use a 4.7µH inductance at the input and at the output.** **Pin 1: +5V** **Pin 2: GND** **Pin 3: isolated GND** **Pin 5: +12 V**
	**VRA0512ZP-6WR3 or VRA0515ZP-6WR3** **https://de.aliexpress.com/item/4000945424164.html** **https://de.aliexpress.com/item/1005003711272767.html** **Input 4.5 V - 9 V** **Output +-12 V 250 mA or +-15 V 200 mA** **The output voltage is stabilized.**

SZ-BK0612 400W DC-DC Step-Down Buck Converter

https://de.aliexpress.com/item/1005004398548528.html
60mm x 60mm x 45mm
Input voltage: 10.5 V - 60 V
Output voltage: 0.15 V - 45 V (Maximum output voltage = 0.8 * Input voltage)
Output current: 0.2-15 A (forced cooling is required above 12 A)
Maximum output power: 400 W (with forced cooling)
Continoous power: 280 W (with normal cooling)
Continuous output current: 12A (with normal cooling)
Input undervoltage protection: Yes, if Input < 10.5 V
Input overvoltage protection: No.
Switching frequency: 100 kHz
Efficiency: up to 97%
Short circuit protection: yes, constant current
One end of the voltage and current potentiometers is unused. That are only variable resistors.

RV2 [kΩ]	0	5	10	15	20	25	30	35	40	45	50	55	60	65	65.9
Voltage [V]	0.15	2.39	4.64	6.90	9.15	11.41	13.66	15.90	18.16	20.4	22.6	24.9	27.1	29.4	30.0

RV1 [kΩ]	0	5	10	15	20	25	30	35	40	50	60	70	80	90
Current [A]	0.08	2.58	4.47	5.95	7.15	8.10	8.86	9.50	10.18	11.20	12.05	12.55	13.03	13.43



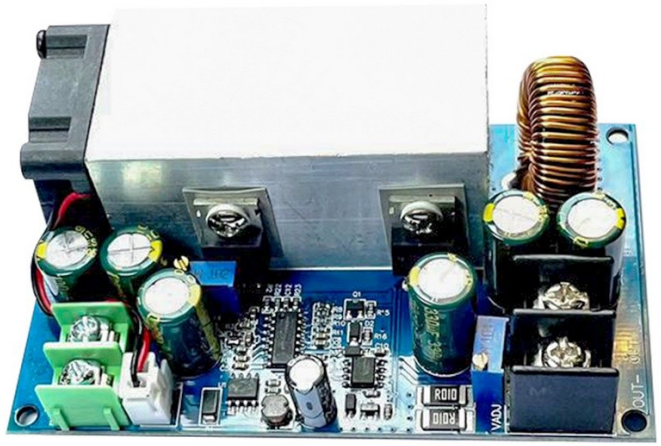
600W DC-DC Step-Down Buck Converter It has no type number printed on the module.

https://de.aliexpress.com/item/1005005240989258.html
100mm x 60mm x 35mm
Input voltage: 12 V - 75 V
Output voltage: 2.5 V - 60 V (Maximum output voltage = 0.8 · Input voltage)
Output current: 25 A
Maximum output power: 600 W (with forced cooling)
Continoous power: 500 W (with normal cooling)
Input undervoltage protection: No.
Input overvoltage protection: No.
Switching frequency: 80 kHz
Efficiency: 95%
Short circuit protection: yes, constant current, but the current limit is not adjustable.

The type numbers of all three SO chips are ground off.

For an adjustable current limit, replace this 1 kΩ resistor (red arrow) by a 1 kΩ potentiometer. The left side is connected to IN-. The potentiometer value is approximately 37 Ω / A.

There exists also a module with an adjustable current limit, see below.



600W DC-DC Step-Down Buck Converter Type number SZ-8025CCCCV or GXJA0758-001 ?

https://de.aliexpress.com/item/1005005440291920.html

This is almost the same module as above, but it has an adjustable current limit 1 A - 25 A.
Problem: The 1 kΩ potentiometer for current adjustment is very close to the power transistor.

The type numbers of all three SO chips are ground off.

Potentiometer RV2 [Ω]	42.5	77	181	367	557
Current limit [A]	1	2	5	10	15

The potentiometer value is approximately 37 Ω / A.

600W DC-DC Step-Down Buck Converter

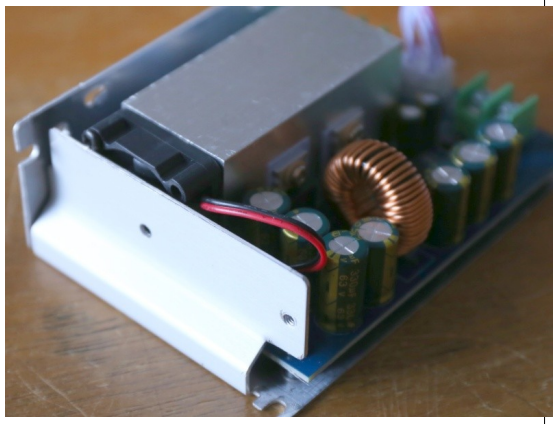
https://de.aliexpress.com/item/1005005967112080.html

Input voltage 12 V - 80 V

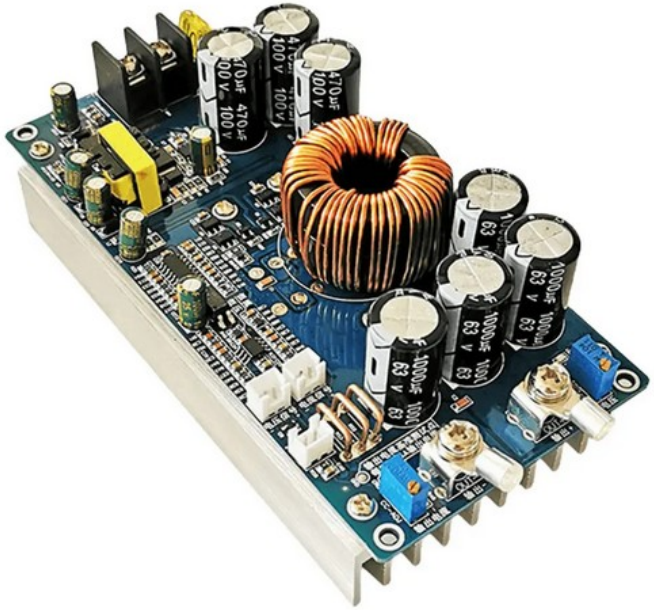
Output voltage 2.5 V - 50 V

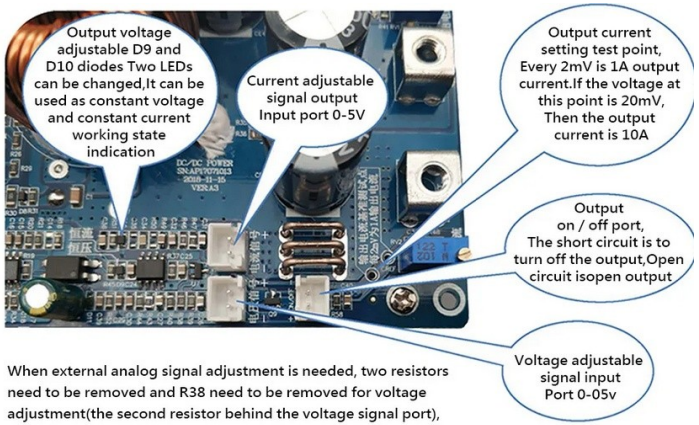
Output current 20 A

Available in two versions, either with potentiometers on the board, or with external potentiometers.



Unbelievable stupid design: The sheet metal is totally blocking the airflow of the fan. It's best if you remove this part of the housing. Also it's better if the fan is blowing in the other direction, so that warm air is not





When external analog signal adjustment is needed, two resistors need to be removed and R38 need to be removed for voltage adjustment(the second resistor behind the voltage signal port), adjust the current and remove R46 (behind the current signal port) At this time, the voltage and current can be adjusted for voltage and current correction respectively.

800W DC-DC Step-Down Buck Converter

https://de.aliexpress.com/item/1005006302567974.html

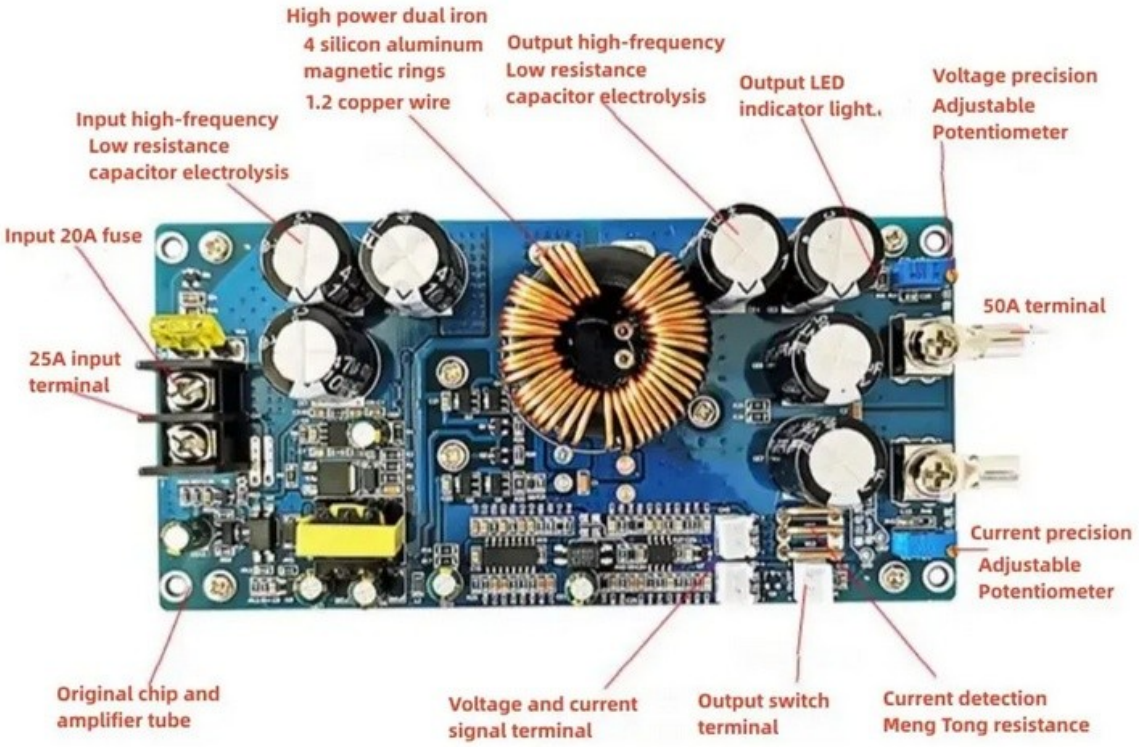
Input voltage 20 V - 70 V

Input current 20 A

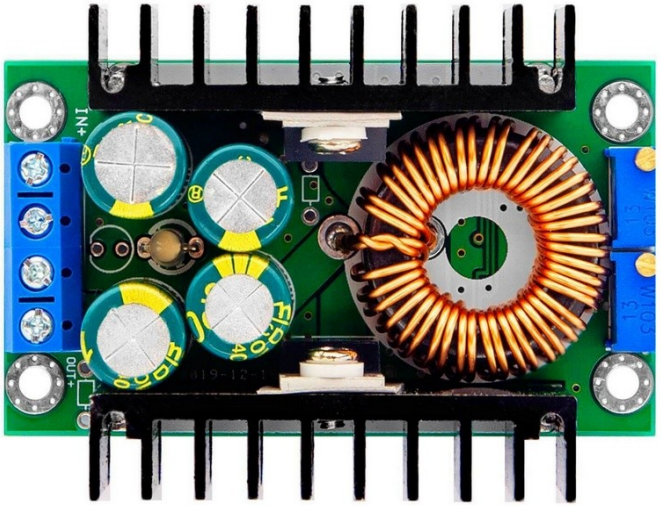
Has input undervoltage protection that switches off the output, if the input voltage is below 20V.

Output voltage 2.5 V - 58 V (100k Potentiometer)

Output current 30 A (1k Potentiometer)

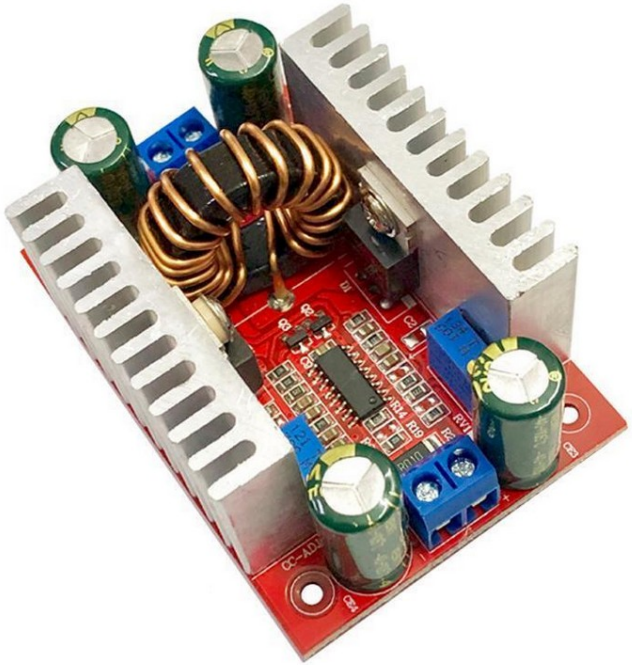
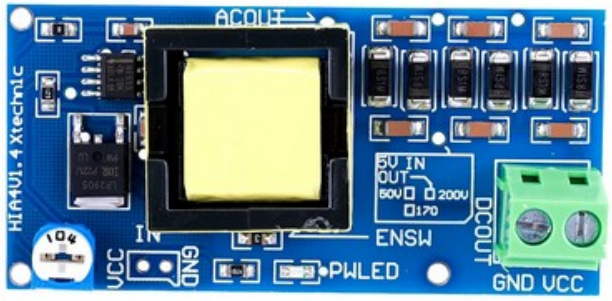


	**LM2596S DC-DC Step-Down Module** **https://de.aliexpress.com/item/1005001621899005.html** **46mm x 23mm x 10mm** **Input voltage: 7V - 35V** **Output voltage: 1.25 V - 30 V** **Difference between input and output voltage: min. 1.5V** **Output current: maximum 3A** **Efficiency: max. 92%** **Switching frequency: 150kHz** **Left potentiometer is voltage, middle potentiometer is switching point for "charge" LED, right potentiometer is current limit.** **Marking on PCB: XW026FR4**
	**LM2596 DC-DC Step-Down Module with Voltmeter** **https://www.ebay.de/itm/125248503976** **61mm · 34mm · 12mm** **Input voltage: 5V - 40V** **Output voltage: 1.5V - 35 V** **Input current: 4A (Max)** **Output current: 2A (Max)** **Protected against wrong polarity and short circuit** **The voltmeter can measure the input or output voltage, or can be switched off.**

	**XH-M404 DC 4–40 V 8 A Module with Voltmeter** **https://www.temu.com/goods.html?_bg_fs=1&goods_id=601099513527641** **Input voltage: 4V - 40V** **Output voltage: 1.25 - 36 V** **Maximum current: 8A (5A continuous)** **Maxumun power: 200W** **Efficiency: 94%** **Short circuit protection, input polarity protection, overtemperature protection.**
	**XL4016 Step Down Buck Converter** **https://www.ebay.de/itm/144127302529** **65mm x 48mm x 24mm** **Input voltage: 5 - 40V** **Output voltage: 1.2 - 35V** **Output current: 4A (forced cooling is required above 65°C)** **Adjustable current limit: 0.2A - 4.0A** **Output power: up to 140W**

	**8A DC-DC Step-Down Module XL4016** **https://www.ebay.de/itm/175521917179** **61mm · 41mm · 27mm** **Input voltage: 4V - 40V** **Output voltage: 1.25V - 36V** **Model number: XH-M401** **Maximum output current: 8A (5A continuous, forced cooling above 5A)** **Maximum power: 200W** **Efficiency: max. 94%** **Switching frequency: 180 kHz** **Problem: The voltage potentiometer might be negative logarithmic? It's difficult to set a small output voltage.**
	**XL7015 DC-DC Converter Step-down Module** **https://de.aliexpress.com/item/1005004584302304.html** **Input voltage: 5 V - 80 V** **Output voltage: 5V - 20V** **Maximum Output current: 0.8 A** **Maximum output power: 7 W** **44 mm x 16 mm** **XL7015E1 datasheet: https://pdf1.alldatasheetde.com/datasheet-pdf/view/1134399/XLSEMI/XL7015E1.html**

	**DC-DC Converter Module DD7818TA** **https://de.aliexpress.com/item/1005003141428190.html** **Input voltage: 8 V - 80 V** **Output voltage: Different modules are available from 3.3 V to 15 V** **Output current: 2.1 A** **60 mm x 30 mm x 16 mm**
	**DC-DC Converter Module DD8024TA** **https://de.aliexpress.com/item/1005005878674288.html** **Input voltage: up to 80 V** **Output voltage: Different modules are available from 3.3 V to 15 V** **Output current: 1 A** **59 mm x 30 mm x 16 mm**

	**400W Step-Up Module** **Input voltage: 8.5 V to 50 V, no protection for wrong input polarity** **Input current: max. 15 A (has a fuse on the board)** **Output voltage: 10 V to 60 V** **Output current limiter: 0.2 A to 12 A (forced cooling required above 7A)** **Switching frequency: 150 kHz** **Dimensions: 67 x 48 x 28 mm** **Left potentiometer in picture is current limit, right potentiometer is voltage** **It seems the current limit can't be set with shorted output.** **The chip is TL494C: https://www.ti.com/lit/ds/symlink/tl494.pdf** **https://www.ebay.de/itm/386264182944** **https://de.aliexpress.com/item/1005003041306288.html**
	**High Voltage DC-DC Boost Converter 5V-12V Step Up To 300V-1200V** **https://de.aliexpress.com/item/1005006858866190.html** **Power: 5W** **Input voltage: 5-12V (less than 12V)** **When the input voltage is 5V, the output voltage can be adjusted from 280 to 1200** **When the input voltage ranges from 6 to 12V, the output voltage ranges from 400 to 1200V** **Product size: 65.00·32.00·16.00mm**

	**8-32V to 45-390V DC-DC Boost Converter Step Up Power Supply Module** **https://www.aliexpress.com/item/1005005927250061.html** **This module is available in two versions, with unipolar or bipolar output.** **Input voltage: 8-32V** **Input current: 5A (MAX)** **Output voltage: 45-390V or +-45-390V continuously adjustable** **Output current: 0.2A MAX** **Output power: 40W (peak 70W)** **Operating frequency: 75KHz** **Conversion efficiency: up to 88%** **Over-current protection: Yes. (Input current exceeds 4.5A, lower output voltage)** **Overvoltage protection: Yes. (Output voltage exceeds 410V, lower output voltage)** **Input reverse protection: Yes (not self-recovery type, then anti-burning fuse, try not to reverse.)**

20W DC-DC Boost-Buck module with dual output Type: DD39AJPA

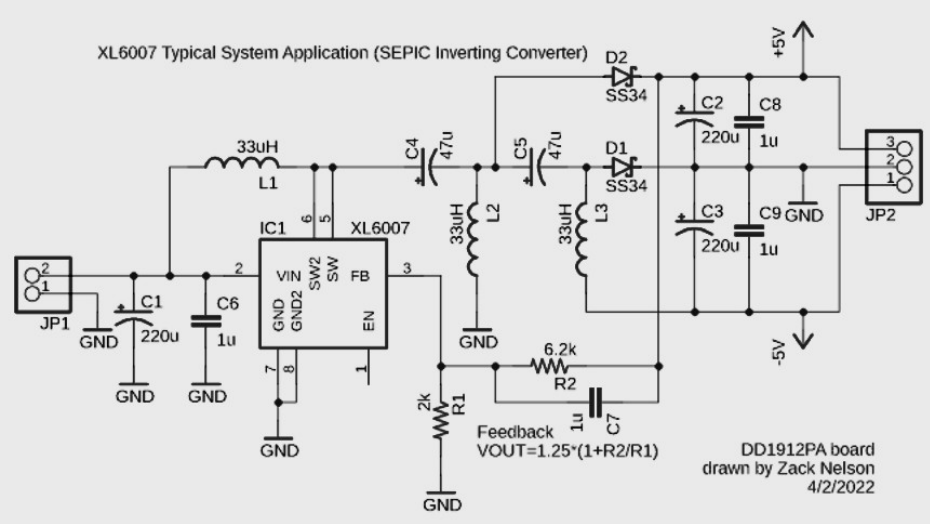
Input voltage 5 V - 30 V

Output voltage +-3 V to +-30 V

https://de.aliexpress.com/item/1005002051989448.html

XL6016: http://www.xlsemi.net/datasheet/XL6019-EN.pdf

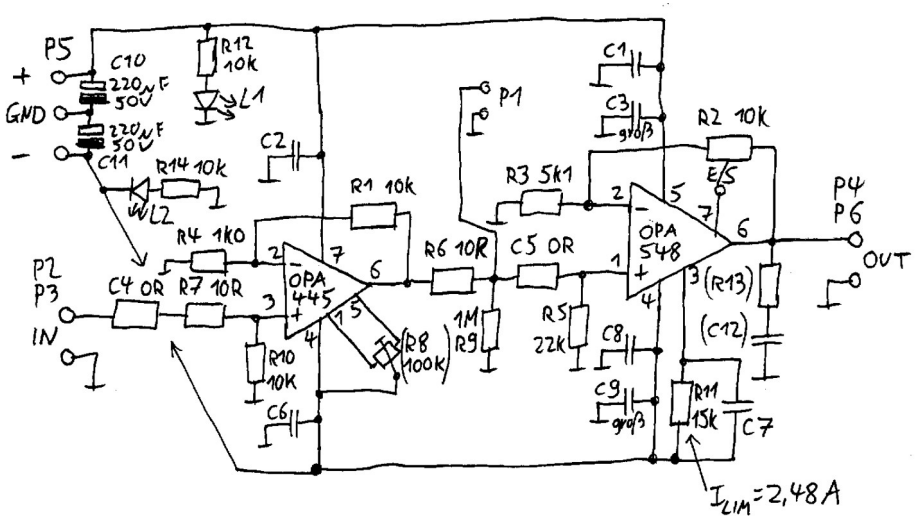
I think the schematic is similar to this one:



18 Chinese Electronic Modules

OPA548 Power-OP-Amp Module

https://de.aliexpress.com/item/32959444043.html
https://de.aliexpress.com/item/1005005046010895.htm
96mm x 64mm x 42mm
Supply voltage: +-10V ... +-30V (OPA548 works with +-4V, but OPA445 needs +-10V, replace by OPA551 for +-4V supply)
Continuous output current: 3A
Peak output current: 5A
Gain-bandwidth product: 1MHz
Has a OPA445 chip as pre-amplifier.
First stage gain: R1 / R4 + 1 = 11 non-inverting
Second stage gain: R2 / R3 + 1 = 2.96 non-inverting
Total gain: 32.57 non-inverting

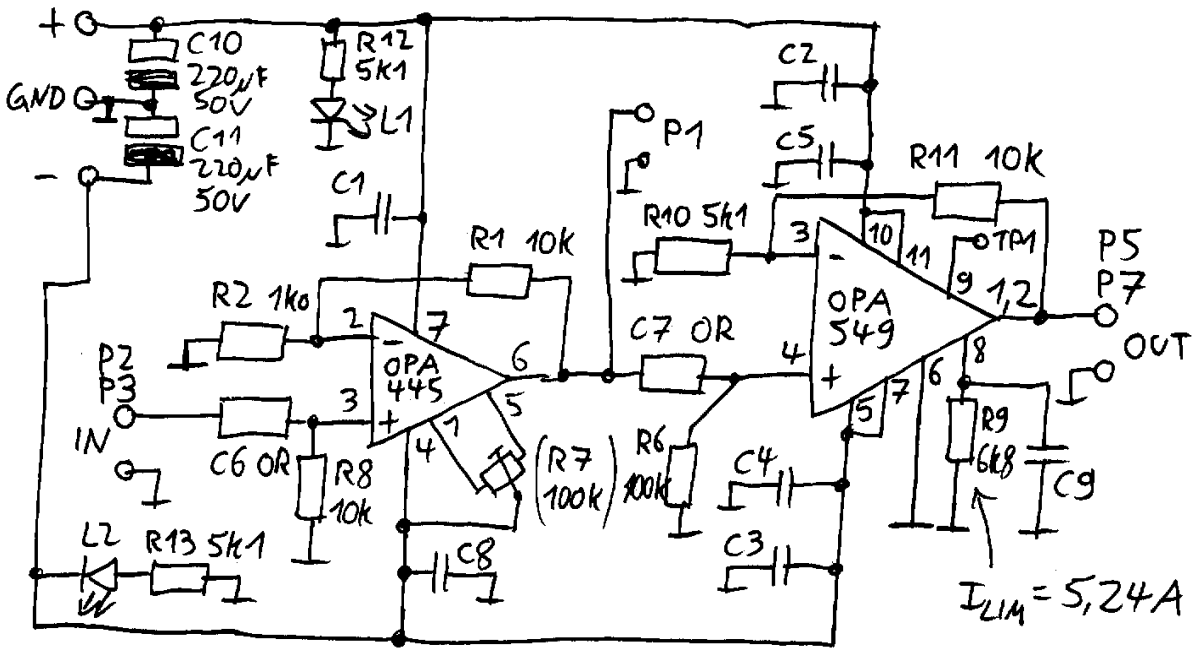


I_{LIM}	R_{CL}
1 A	67.5 kΩ
2 A	30 kΩ
3 A	17.5 kΩ
4 A	11.25 kΩ
6 A	5 kΩ
8 A	1.875 kΩ

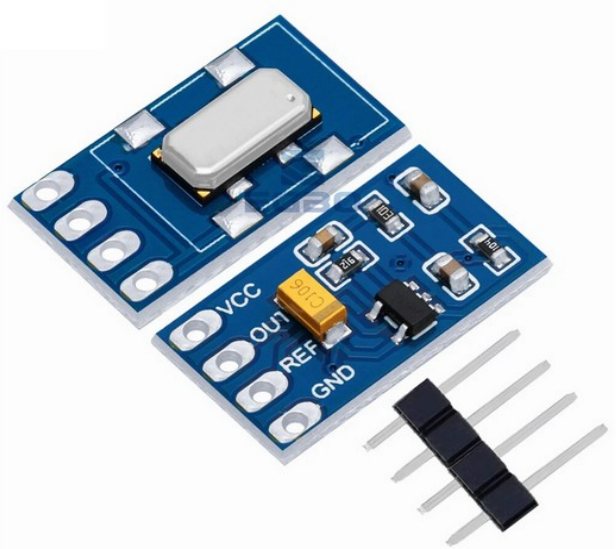
OPA549 Power-OP-Amp Module

https://de.aliexpress.com/item/1005001876749797.html
https://de.aliexpress.com/item/1005005046010895.html

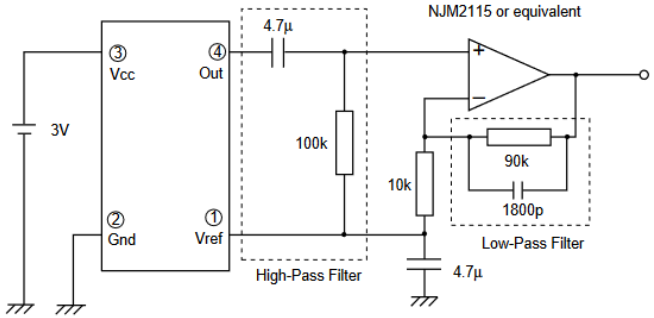
86mm x 81mm x 52mm
Supply voltage: +-10V ... +-30V
It would work from +-4V to +-30V, if the preamplifier OPA445 is replaced by OPA551
Continuous output current: 8A Peak output current: 10A
Gain-bandwidth product: 900kHz Input impedance: 10 kOhm Input voltage: 2Vpp (max)
Output voltate: 54 Vpp (max) with +-30 V supply Current limit is set to 5.24 A by default
Output power: 100 W (max)
Gain: first stage 11, second stage 3, total 33, g1 = r1/r2 + 1, g2 = r11/r10 + 1
Input offset voltage: 1mv (max) Input offset drift: 2 uV/°C (max)
Problem: There is an air gap between the OPA549 and the heatsink, and when you correct this by adjusting the heatsink position, then two of the mounting nuts are very close to the heatsink.



	**OPA549 Power-OP-Amp Module with large Heatsink (there may be two different version available, constant voltage and constant current, I'm not sure)** https://de.aliexpress.com/item/1005005957798467.html https://de.aliexpress.com/item/1005005790167035.html https://de.aliexpress.com/item/1005004654749888.html https://de.aliexpress.com/item/1005005957420987.html https://de.aliexpress.com/item/1005003740927088.html



This is the sample circuit from the ENC-03R datasheet:



The hiph-pass filter's cut-off frequency in this circuit is approx. 0.3Hz.
The low-pass filter's cut-off frequency in this circuit is approx. 1kHz.

GY-35-RC Single-axis gyroscope module

https://de.aliexpress.com/item/1005005997583188.html

Chip: ENC-03RC (this chip is discontinued)
Datasheet : https://www.elecrow.com/download/ENC-03.pdf

According to the datasheet, the sensor should be marked as „C416“, but it is marked as „C234“.

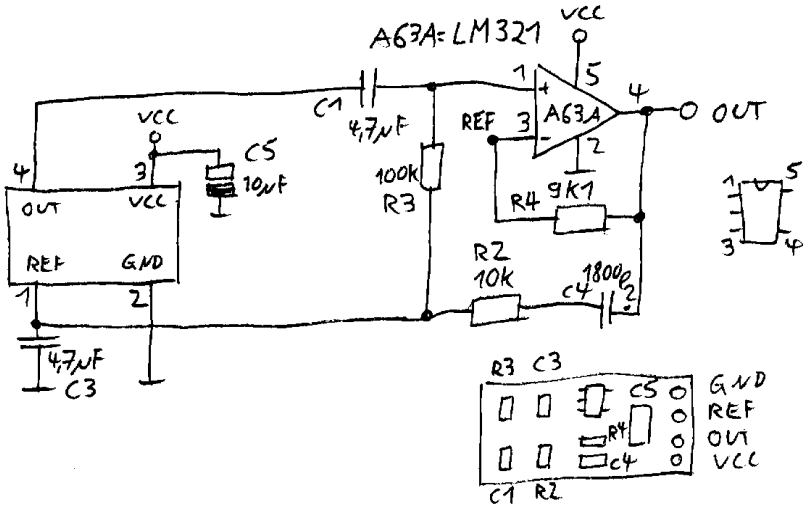
Power supply: 2.7 V - 5.25 V

Scale factor in datasheet: 0.67mV/°/s but that's wrong! It must be 0.67mV/(°/s) or 0.67mVs/°

Maximum Angular Velocity: +-300 °/s (which is equivalent to +-201mV)

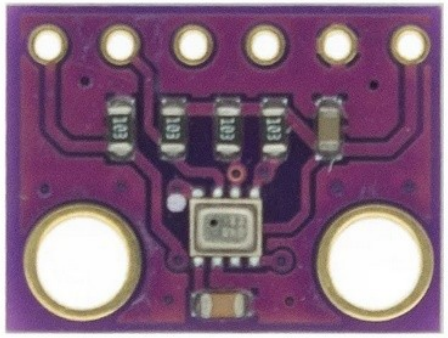
Op-Amp: LM321 (not MPC601 as in the description on Aliexpress, the pinout is different!)

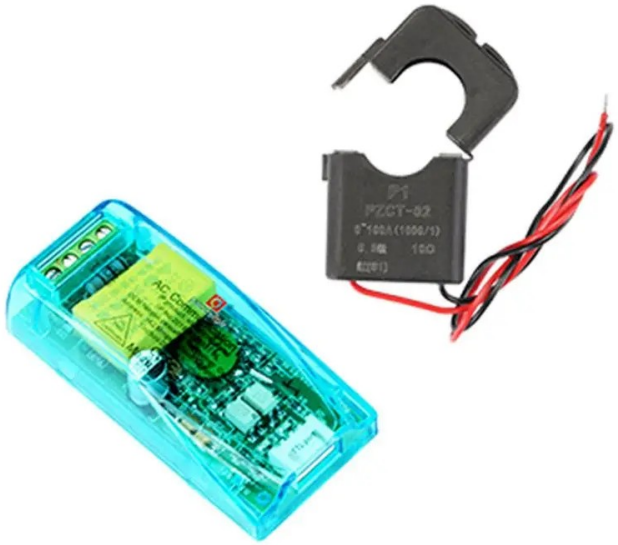
This is the schematic diagram of the board, after reverse-engineering. It has many problems.
To fix it, replace the 9k1 resistor by a 90K resistor (or two parallel 180k resistors), and add the missing connection from from R2,C4 to the inverting input of the Op-Amp.
Also the REF output should be disconnected from the inverting input of the Op-Amp and connected to R2,R3,C3.
I'm unsure it REF is an input or aoutut.



Don't but this chinese crap. Even after these changes, it doesn't work. The output signal of the sensor makes no sense at all. I'm not sure if the sensor is really ENC-03RC.

See also:https://www.makerfabs.com/1-axis-analog-gyro-module-enc03.html
https://wiki.mikrokopter.de/GyroScope

	**Digital Pressure Sensor BMP280** **https://de.aliexpress.com/item/1005006585367042.html** **Datasheet: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf**



PZEM-004T AC 100A Digital Ampere / Volt/ Watt measurement module with RS485

Available as a 10A module (with internal shunt) or as a 100A module (with current clamp, either open or closed).

The current clamp is a 1000:1 current transformer and the output load must be 10 Ohm or less.

In this module the isolation is made by optocouplers between the microcontroller and the RS485 output. The microcontroller is on mains voltage level, and it's powered by a capacitor from the mains voltage (seems risky if the mains voltage contains spikes). But you also need 5V supply at the output side, for the optocoupler. It can be modified to 3.3V by changing a resistor.

https://www.aliexpress.com/item/1005004852373322.html

Arduino library: https://github.com/mandulaj/PZEM-004T-v30

Unfortunately, programming is very complicated because a CRC check is required for all data transmissions. You have to rely on the Arduino Library, without understanding how CRC works.

Wiki: https://github.com/olehs/PZEM004T/wiki

This page contains useful instructions how to test the module with a PC software:
https://www.nn-digital.com/en/blog/2019/11/04/example-of-the-pzem-004t-v3-v3-0-interfacing-program-using-arduino/
Extract the files to a folder, then start "run.bat" as administrator, which will register a library. Then run "PZEM004T-Master.exe". This test was successful.

The Arduino library mentioned in the above website is not suitable for SAMD21 microcontroller.
A modified library for SAMD21 exists here: https://github.com/zygisjas/PZEM-004T-V30-SAMD21/tree/main
But I didn't succeed to install it in Arduino IDE. This Github stuff is all a big mess.

Baud rate is 9600, 8 data bits, 1 stop bit, no parity

Modification for measuring lower line voltage (from a variable transformer):
https://www.mikrocontroller.net/attachment/315769/Eingangsschaltung_zum_Display.pdf

2.2 Application layer protocol
The application layer use the Modbus-RTU protocol to communicate. At present, it only supports function codes such as 0x03 (Read Holding Register), 0x04 (Read Input Register), 0x06 (Write Single Register), 0x41 (Calibration), 0x42 (Reset energy).etc.0x41 function code is only for internal use (address can be only 0xF8), used for factory calibration and return to factory maintenance occasions, after the function code to increase 16-bit password, the default password is 0x3721
The address range of the slave is 0x01 ~ 0xF7. The address 0x00 is used as the broadcast address, the slave does not need to reply the master. The address 0xF8 is used as the general address, this address can be only used in single-slave environment and can be used for

calibration etc. operation.

2.3 Read the measurement result
The command format of the master reads the measurement result is(total of 8 bytes):Slave Address + 0x04 + Register Address High Byte + Register Address Low Byte + Number of Registers High Byte + Number of Registers Low Byte + CRC Check High Byte + CRC Check Low Byte.

The command format of the reply from the slave is divided into two kinds:
Correct Reply:
Slave Address + 0x04 + Number of Bytes + Register 1 Data High Byte + Register 1 Data Low Byte + ... + CRC Check High Byte + CRC Check Low Byte
Error Reply:
Slave address + 0x84 + Abnormal code + CRC check high byte + CRC check low byte

Abnormal code analyzed as following (the same below)
- 0x01,Illegal function
- 0x02,Illegal address
- 0x03,Illegal data
- 0x04,Slave error

The register of the measurement results is arranged as the following table

Register address	Description	Resolution
0x0000	Voltage value	1LSB correspond to 0.1V
0x0001	Current value low 16 bits	1LSB correspond to 0.001A
0x0002	Current value high 16 bits	
0x0003	Power value low 16 bits	1LSB correspond to 0.1W
0x0004	Power value high 16 bits	
0x0005	Energy value low 16 bits	1LSB correspond to 1Wh
0x0006	Energy value high 16 bits	
0x0007	Frequency value	1LSB correspond to 0.1Hz
0x0008	Power factor value	1LSB correspond to 0.01
0x0009	Alarm status	0xFFFF is alarm, 0x0000is not alarm

For example, the master sends the following command (CRC check code is replaced by 0xHH and 0xLL, the same below) 0x01 + 0x04 + 0x00 + 0x00 + 0x00 + 0x0A + 0xHH + 0xLL Indicates that the master needs to read 10 registers with slave address 0x01 and the start address of the register is 0x0000
The correct reply from the slave is as following: 0x01 + 0x04 + 0x14 + 0x08 + 0x98 + 0x03 + 0xE8 + 0x00 + 0x00 +0x08 + 0x98+ 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x01 + 0xF4 + 0x00 + 0x64 + 0x00 + 0x00 + 0xHH + 0xLL
The above data shows
- Voltage is 0x0898, converted to decimal is 2200, display 220.0V
- Current is 0x000003E8, converted to decimal is 1000, display 1.000A
- Power is 0x00000898, converted to decimal is 2200, display 220.0W
- Energy is 0x00000000, converted to decimal is 0, display 0Wh

- Frequency is 0x01F4, converted to decimal is 500, display 50.0Hz
- Power factor is 0x0064, converted to decimal is 100, display 1.00
- Alarm status is 0x0000, indicates that the current power is lower than the alarm power threshold

2.4 Read and modify the slave parameters
At present,it only supports reading and modifying slave address and power alarm threshold
The register is arranged as the following table

Register address	Description	Resolution
0x0001	Power alarm threshold	1LSB correspond to 1W
0x0002	Modbus-RTU address	The range is 0x0001~0x00F7

The command format of the master to read the slave parameters and read the measurement results are same(descrybed in details in Section 2.3), only need to change the function code from 0x04 to 0x03.
The command format of the master to modify the slave parameters is (total of 8 bytes):
Slave Address + 0x06 + Register Address High Byte + Register Address Low Byte + Register Value High Byte + Register Value Low Byte + CRC Check High Byte + CRC Check Low Byte.
The command format of the reply from the slave is divided into two kinds:
Correct Response: Slave Address + 0x06 + Number of Bytes + Register Address Low Byte + Register Value High Byte + Register Value Low Byte + CRC Check High Byte + CRC Check Low Byte.
Error Reply: Slave address + 0x86 + Abnormal code + CRC check high byte + CRC check low byte.
For example, the master sets the slave's power alarm threshold:
0x01 + 0x06 + 0x00 + 0x01 + 0x08 + 0xFC + 0xHH + 0xLL Indicates that the master needs to set the 0x0001 register (power alarm threshold) to 0x08FC (2300W).Set up correctly, the slave return to the data which is sent from the master.For example, the master sets the address of the slave
0x01 + 0x06 + 0x00 + 0x02 + 0x00 + 0x05 + 0xHH + 0xLL
Indicates that the master needs to set the 0x0002 register (Modbus-RTU address) to 0x0005 Set up correctly, the slave return to the data which is sent from the master.

2.5 Reset energy
The command format of the master to reset the slave's energy is (total 4 bytes):Slave address + 0x42 + CRC check high byte + CRC check low byte.Correct reply: slave address + 0x42 + CRC check high byte + CRC check low byte.
Error Reply: Slave address + 0xC2 + Abnormal code + CRC check high byte + CRC check low byte

2.6 Calibration
The command format of the master to calibrate the slave is (total 6 bytes):0xF8 + 0x41 + 0x37 + 0x21 + CRC check high byte + CRC check low byte. Correct reply: 0xF8 + 0x41 + 0x37 + 0x21 + CRC check high byte + CRC check low byte.Error Reply: 0xF8 + 0xC1 + Abnormal code + CRC check high byte + CRC check low byte.It should be noted that the calibration takes 3 to 4 seconds, after the master sends the command, if the calibration is successful, it will take 3 ~ 4 seconds to receive the response from the slave.

2.7 CRC check
CRC check use 16bits format, occupy two bytes, the generator polynomial is X16 + X15 + X2 +1, the polynomial value used for calculation is 0xA001.The value of the CRC check is a frame data divide all results of checking all the bytes except the CRC check value.

Voltage Range: AC 85-300V Current Range: AC 20mA-50A Power Supply: DC 3.3V Power Consumption: ≤10mA Baud Rate: Default 9600bps Data Format: N,8,1 Mailing address: Default No.1 	**JSY1003F Single Phase Embedded AC Acquisition Module** **In this module the isolation is between the mains voltage and the microcontroller. The microcontroller is powered by an external 3.3V source. Much better solution!** **Data sheet: https://www.jsypowermeter.com/uploads/JSY1003F-User-Manual1.pdf** **https://www.jsypowermeter.com/jsy1003f-single-phase-ac-ttl-modbus-rtu-embedded-metering-module-product/** **https://de.aliexpress.com/item/4000340221120.html** **The data protocol is different from the PZEM-0004T module: Different module adress, different register adresses.** **The ADC chip is a RN8208G, which has a programmable gain in front of the converter. The datasheet seems to be available only in chinese: https://www.lcsc.com/datasheet/lcsc_datasheet_2111261730_RENERGY-RN8208G_C2925739.pdf** **Pinout from left to right: L nc nc N CT+ CT- GND 3V3 RX TX PF IRQ**
Voltage Range: AC 1-380V Current Range: AC 5mA-16A Power Supply: DC 5V/3.3V Power Consumption: ≤10mA Baud Rate: Default 9600bps Data Format: N,8,1 Mailing address: Default No.1 	**JSY1017 Single Phase Embedded AC Acquisition Module** **In this module the isolation is between the mains voltage and the microcontroller. The microcontroller is powered by an external 3.3V source. Much better solution!** **Data sheet: https://www.jsypowermeter.com/uploads/JSY1017-User-Manual.pdf** **https://de.aliexpress.com/item/1005006724544182.html**

	**2000W Soft Start Board 30A** **https://www.aliexpress.com/item/1005002907433282.html** **Has a pushbutton for on / off, and two inputs for overtemperature switches (normal open)** **The black module is a small switching power supply with 12V 250mA output.** **One relay is for on / off, the other relay is for bypassing the NTCs.** **Dimensions: 96 mm x 82 mm**
	**2000W Soft Start Board 30A** **https://www.aliexpress.com/item/1005002906542286.html** **Dimensions: 74 mm x 57 mm**

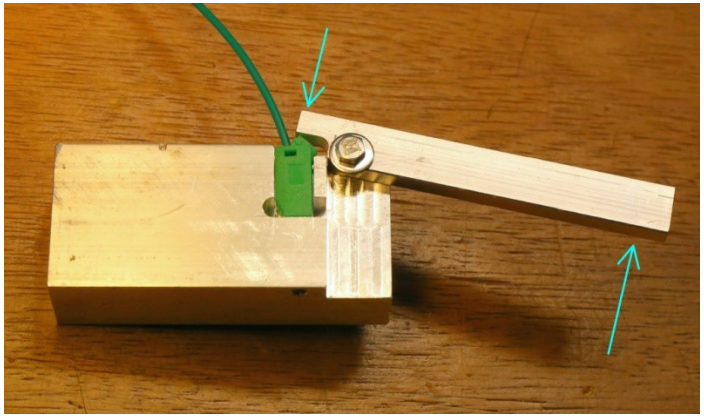
	**1500W Soft Start Board** **https://www.aliexpress.com/item/32844604900.html** **Dimensions: 69 mm x 67 mm**
	**5000W Soft Start Board** **https://www.aliexpress.com/item/1005006046571334.html** **Dimensions: 84 mm x 63 mm**

Switching Power Supply Modules 3W

https://www.aliexpress.com/item/32919781320.html

Input voltage: 100V - 240V AC

Type	Output
PM03	3.3 V 1.0 A
PM01	5 V 0.6 A
PM09	9 V 0.33 A
PM12	12 V 0.25 A
PM15	15 V 0.2 A
PM24	24 V 0.125 A



Left side: Tool for closing the 2mm plugs

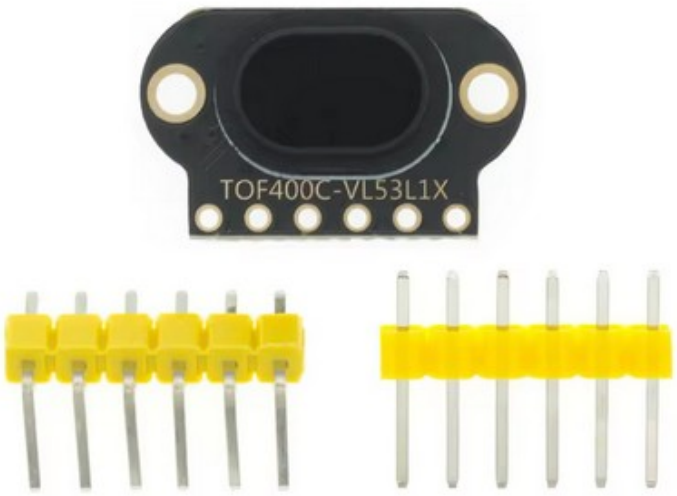
2mm Plugs:

https://de.aliexpress.com/item/926259246.html

2mm Jacks:

https://de.aliexpress.com/item/4000271665708.html

https://de.aliexpress.com/item/32816281597.html



Time-of-Flight Module TOF400C-VL53L1

Datasheet: https://www.st.com/resource/en/datasheet/vl53l1.pdf

Description of the fields of view: https://www.st.com/resource/en/application_note/an5894-description-of-the-fields-of-view-of-stmicroelectronics-timeofflight-sensors-stmicroelectronics.pdf

If you are using version 1.6.2 or later of the Arduino software (IDE), you can use the Library Manager to install this library: https://github.com/pololu/vl53l1x-arduino

In the Arduino IDE, open the "Sketch" menu, select "Include Library", then "Manage Libraries...".
Search for "VL53L1X". Click the VL53L1X entry in the list. Click "Install".

```
Seeed XIAO SAMD21   VL53L1X board
-----------------   -------------
Pin 12  3V3         VIN
Pin 13  GND         GND
Pin 5   SDA         SDA
Pin 6   SCL         SCL
Pin 1   DAC Output

I2C Adress: 0x52
```

19 List of Abbreviations

Copied from the REDAC manual, sorted alphabetically, and some more abbreviations added.

AC: Alternating Current

ACN: Analog Compute Node

AD: Analog Digital

ADC: Analog to Digital Converter

AMP: (Operational) Amplifier

C-Block: Coefficient Block

CLK: Clock Signal in Serial Interfaces (e.g., SPI)

CM: Common Mode

CMOS: Complementary Metal-Oxide-Semiconductor

CMP: Comparator

CN: Compute Node

CNVST: Conversion Start Signal of an ADC

CS: Chip Select (SPI Signal Line)

CU: Control Unit

DA: Digital Analog

DAC: Digital to Analog Converter

DC: Direct Current

DC/DC: DC-to-DC Converter (Power Supply)

DIMM: Dual Inline Memory Module (Form factor for PCBs)

DDA: Digital Differential Analyzer

DDR: Double Data Rate

DMA: Direct Memory Access

DNL: Differential Nonlinearity

DPDT: Double Pole Dual Throw

DPST: Double Pole Single Throw

DPT: Digital Potentiometer

FDT: Feedthrough

FlexIO: Flexible Input and Output Peripheral (NXP)

GBP: Great Britain Pound

GBW: Gain Bandwidth (Product)

GND: Ground

GPIO: General Purpose Input and Output Pins/Ports

HALT: Halting Mode

HC: Hybrid Controller

I-Block: Current Block

IC: Integrated Circuit or Initial Condition (depending on context)

INL: Integral Nonlinearity

INT: Integrator

INV: Inverter

LSB: Least Significant Bit

M-Block: Math Block

MDAC: Multiplying Digital to Analog Converter

MISO: Master In, Slave Out (SPI Data Line)

Model-1: Analog Paradigm Model-1 Computer

MOSI: Master Out, Slave In (SPI Data Line)

MPDE: Multi-Time Partial Differential Equation

MULT: Multiplier

MUX: Multiplexer

NMOS: N-type Metal-Oxide-Semiconductor

NP: Nondeterministic Polynomial (Complexity Context)

ODE: Ordinary Differential Equation https://en.wikipedia.org/wiki/Ordinary_differential_equation

OP: Operating Mode

OPA: Operational Amplifier

OPV: Operational Amplifier

P: Polynomial (Complexity Context)

PCB: Printed Circuit Board (Electronic Circuit Board)

PDE: Partial Differential Equation https://en.wikipedia.org/wiki/Partial_differential_equation

pDGL: Partial Differential Equation (German: Partielle Differentialgleichung)

PMOS: P-type Metal-Oxide-Semiconductor

PPV: Perturbation Projection Vector

PT1: First Order System with One Pole

PWM: Pulse Width Modulation

REDAC: REConfigurable Analog Computer

RRI: Rail to Rail Input

RRIO: Rail to Rail Input and Output

RRO: Rail to Rail Output

SMD: Surface Mount Device

SOIC: Small Outline Integrated Circuit

SPDT: Single Pole Dual Throw

SPST: Single Pole Single Throw

SPI: Serial Peripheral Interface

SR: Slewrate

T-Block: Topology Block

TSSOP: Thin Shrink Small Outline Package

U-Block: Voltage Block

VDD: Drain Power Voltage

VSSOP: Very Thin Shrink Small Outline Package

XBAR: Crosspoint-switch / Crossbar-switch

XTALK: Crosstalk

Note: Due to a bug in Libre Office / PDF export, the numbers in the index may be wrong by a few pages.

Index