All Kinds of Oscillators

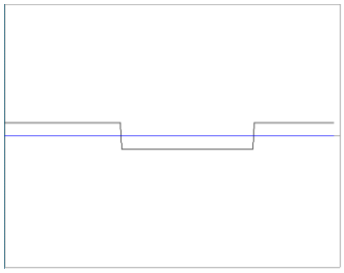
Just about every beginning programming example includes an LED blink routine. There will be no exceptions here. However, expect a lot of variety.

The square wave generator is the easiest way to produce a repetitive square wave. We will use one to make the LED connected to output D25 blink. Then, we will look at the accuracy of the timing.

Type in and run the following program:

```
clr
swg tw p
out p LED
prf p
end
set tw 1
set max 1000
set p 1
set LED 25
```

The built in LED on the Pico will flash on one second then off one second. The output is also printed so you can look at the waveform.



Next, stop, set tw to 0.06 and run the program. It will print six ones then six minus ones. Stop, set tw to 0.07 and run the program. It will print eight ones then eight minus ones. It is running one dt step or 0.01 seconds slower. The square wave generator accumulates time each tw time width as a floating point number. Once roundoff error makes the accumulated time slightly less than tw, it takes another dt cycle to switch states. Setting tw slightly smaller at 0.0699 compensates for the roundoff error.

tw = 0.06

```
-1.000000
-1.000000
-1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
1.000000
1.000000
```

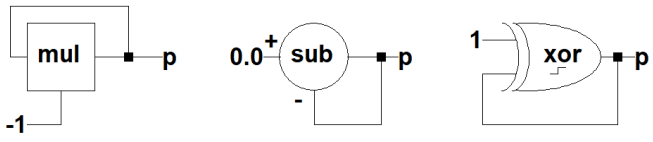
tw = 0.07

```
1.000000
1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
-1.000000
```

tw = 0.0699

```
-1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
-1.000000
1.000000
1.000000
```

Several types of positive feedback systems can be made to toggle at each dt step. Here are some examples:



Multiplier with a -1 to +1 toggle. For a -N to +N toggle, set p to value N:

```
clr
mul p -1 p
out p LED
prf p
end
set -1 -1
set max 1000
set p 1
set LED 25
set dt 1
```

Subtraction with a -3 to +3 toggle (or any starting value):

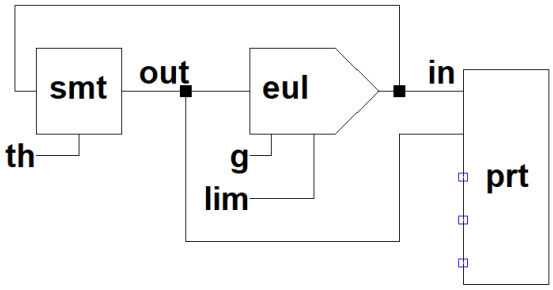
```
clr
sub 0 p p
out p LED
prf p
end
set max 1000
set p 3
set LED 25
set dt 1
```

Exclusive OR gate 0 to 1 toggle:

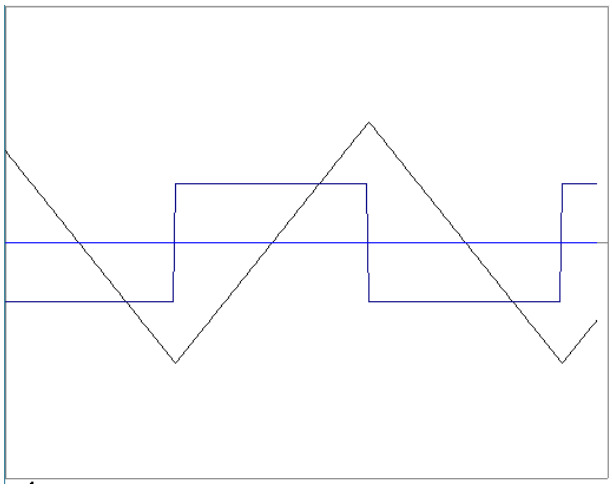
```
clr
xor 1 p p
out p LED
prf p
end
set max 1000
set 1 1
set LED 25
set dt 1
```

The next type of oscillator relies on hysteresis to generate a square wave and a triangle wave. The Schmidt Trigger block has an output that changes between plus and minus one. When it is one, the input must exceed the positive threshold +th value before changing to a negative one. Similarly, when the output is minus one, the input must drop below a negative threshold -th before it switches to a plus one. The Schmidt Trigger inverts the signal but with hysteresis.

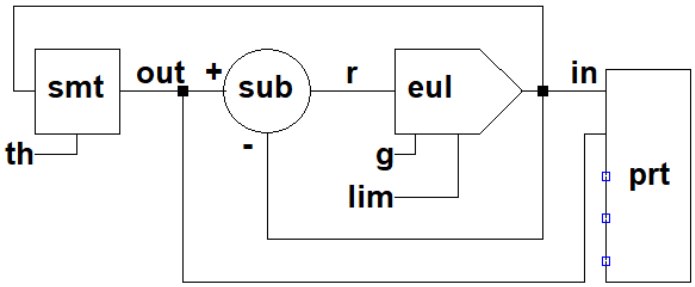
Attaching an integrator to the output will cause the integrator to ramp up when the output is one. Since it is attached to the input of the Schmidt Trigger, it will eventually reach the +th value. Then the output switches to a negative one and the integrator ramps negative to the -th value.



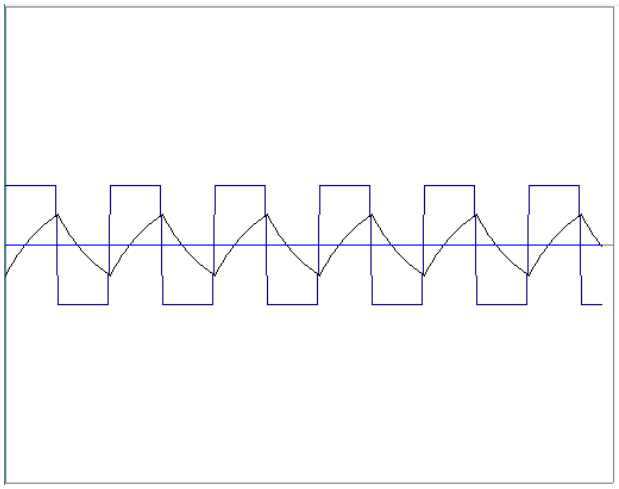

```
clr
smt in th out
eul out g lim in
prt in out
end
set th 2
set out 1
set dt .01
set max 100
set g 5
```



This hysteresis type of oscillator is more often used with a resistor capacitor filter as opposed to the integrator. The common name is a relaxation oscillator. There is no block for a resistor or a capacitor but we can simulate one with negative feedback and an integrator. The RC simulation part has an exponential rise to the out value of one. Therefore, we must keep the th value below one for the system to work.



clr
smt in th out
sub out in r
eul r g lim in
prt in out
end
set th 0.5
set out 1
set dt .01
set max 100
set g 5



The time constant for the RC filter is RC=1/g. The RC signal swings between plus and minus 0.5 and the relative input swing is 1.5. Solving: $1 - 1.5e^{-tg} = 0.5$ gives us the half cycle time t = 1.098/g. For our system, the half cycle is 219.7ms and the period is 439.4ms.

To generate non-square pulses, conditional branches are used. The method requires keeping track of two counts: one for the high level and one for the low level. After the end of cycle is reached the counter is reset.

Non-square pulses are commonly used in pulse width modulation. The on time represents the average voltage value of the signal. Here is an example to modulate the D25 built-in LED. It will use the time t as the brightness level. Time is reset after one second to start over.

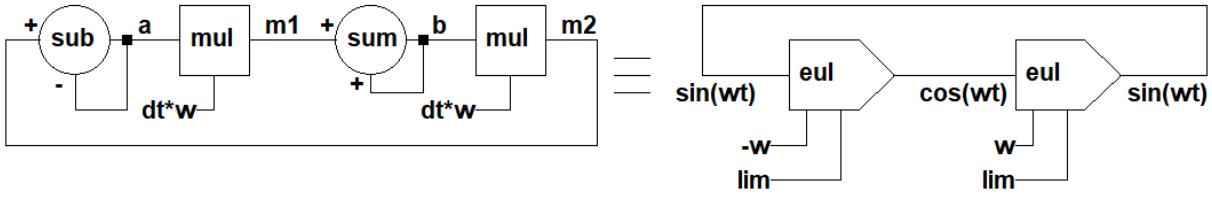
```
clr
sum n .01 n
sub n x d
brm d 2
out L 25
sub n 1 d
brm d 4
out H 25
rst 0 n
rst t x
sub t 1 d
brm d 2
rst 0 t
end
set 1 1
set 2 2
set 4 4
set 25 25
set max 2
set dt 0.0001
set .01 .01
set H 1
set L 0
```

Increment n. If more than x: output a low

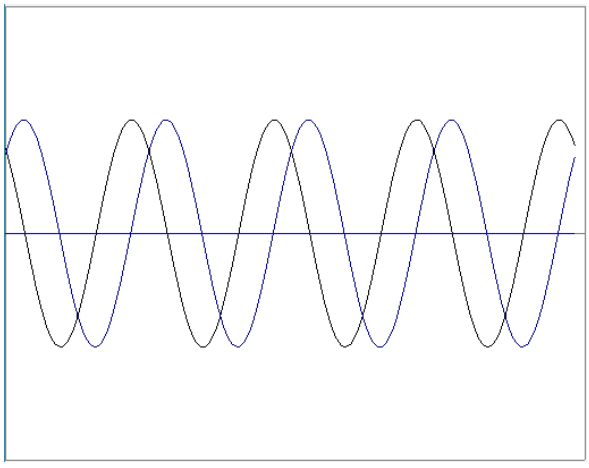
If n is more than 1: output a high, n=0, t=0, x = t

The osc block is locked on the system time t. It produces a sine wave that is in phase from time t=0. Free running sine wave oscillators can be made using two integrators. However, these oscillators require limiting to prevent exponential growth in amplitude.

It is possible to make a two phase quadrature oscillator with just addition and multiplication. The oscillator is always stable in amplitude. It is a two integrator oscillator but the sequencing of the blocks makes one integrator a first order and the other a second order. Magnitude is set with initial conditions and frequency with gain. Start with b=0 and a=magnitude. g = ω*dt



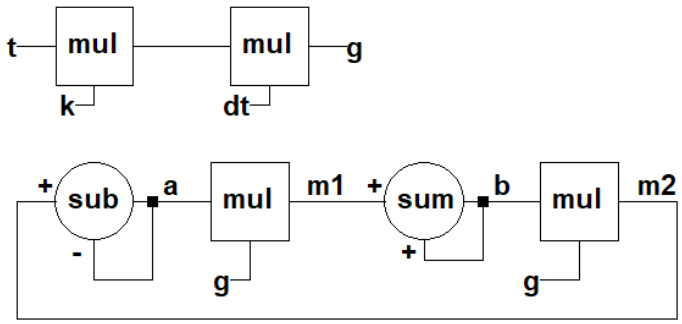
clr
sub a m2 a
mul a g m1
sum m1 b b
mul b g m2
prt a b
end
set dt 0.005
set max 500
set a 1
set b 0.0
set g 0.1



ω=0.1/.005=20rps

The osc block function has a cleaner output with the 3rd harmonic down about 55dB. This oscillator has a 3rd harmonic down about 45 dB from the fundamental.

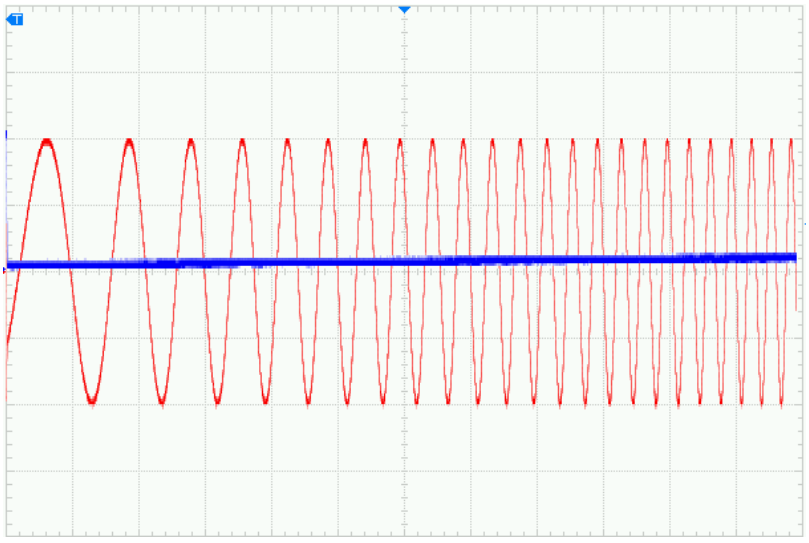
If an increasing variable such as t is used for the frequency value, then the two integrator oscillator will act as a sweep generator. Start with t=1 and end on 10. Set k=ω=2πf. Use a conditional to reset t to one after reaching ten.




```
clr
mul t k tk
mul tk dt g
sub a m2 a
mul a g m1
sum m1 b b
mul b g m2
prf a
sub t 10 td
brm td 2
rst 1 t
end
set dt 0.001
set max 500
set a 1
set b 0.0
set k 62.8
set t 1
set 10 10
set 1 1
set 2 2
```

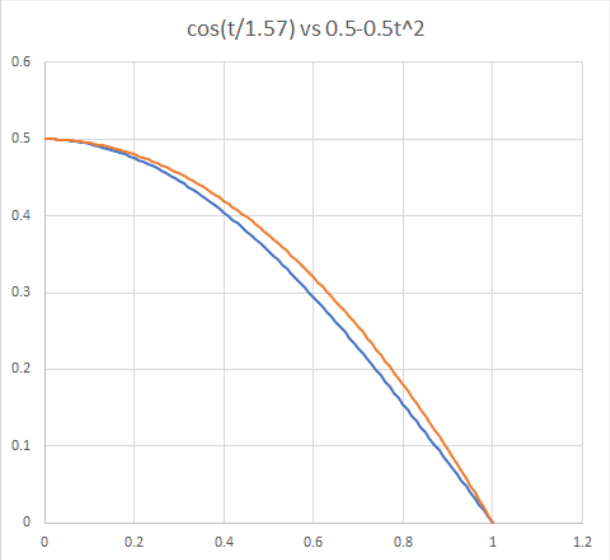
Here is a DAC output version that sweeps 10Hz to 1KHz every ten seconds.

```
clr
mul t k tk
mul tk dt g
sub a m2 a
mul a g m1
sum m1 b b
mul b g m2
dac a ch0
sub t 10 td
brm td 2
rst .1 t
end
set dt 0.00005
set max 500
set a 1
set b 0.0
set k 628
set t .1
set 10 10
set .1 .1
set 2 2
```



Above is a partial o'scope capture of the sweep generator function.

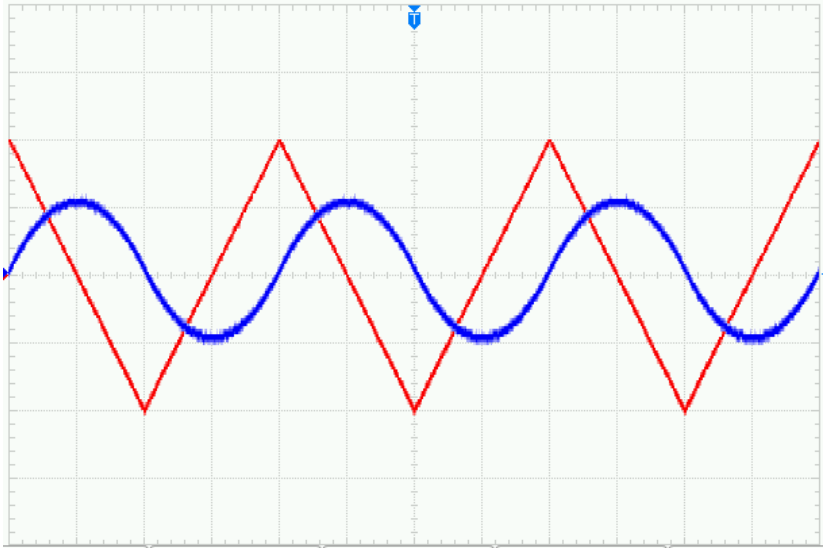
Next, we will examine generating sine curves from a triangle waveform. The first method uses integration. The triangle ramp is an integration of a constant and the integration yields a function proportional to time t. Integrate once more and it yields a function related to $t^2/2$. We can approximate a normalized cosine function with an input zero to one with a constant minus t^2. It turns out $0.5-0.5t^2$ is very close to 0.5cos(t/1.57).



The triangle wave has a positive and a negative slope but we only need to start the integrator at the right value of -0.5 and the integrator will keep track of the rest.

The program below is the Schmidt Trigger triangle wave generator with an added integrator output. It sends a 25Hz triangle wave and integrator output to the two DACs.

```
clr
smt in th out
eul out g lim in
eul in g lim sin
dac in ch0
dac sin ch1
end
set th 0.99
set out 1
set dt .0001
set max 1000
set g 100
set ch1 1
set sin -.5
```



The sine output is not as clean as others. The third harmonic is down about 28dB from the fundamental. The output was scaled to 5V for harmonic testing.

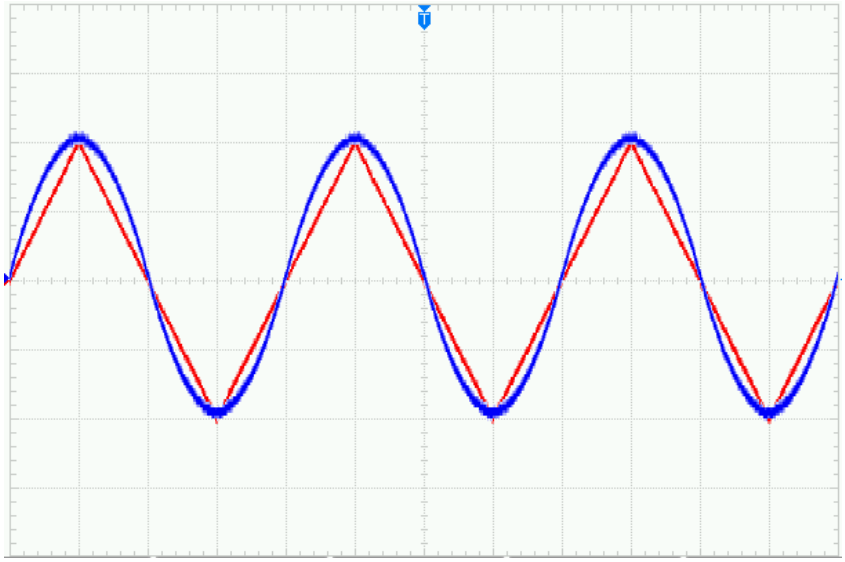
The second method uses a partial Taylor series approximation. Actually, the fit is a second order polynomial minus the constant offset. Below is a fit to a normalized input to a sine function.

The equation needed is $1.875x-0.833x^2$. Squaring the triangle wave input always yields a positive result. Thus, we need to keep track of signs when the triangle wave is negative.

Finally: When thinking about the way the two sine generating methods work, method-1 is a function of time and method-2 is a function of amplitude.

Here is the modified DAC output triangle wave oscillator using the sine polynomial approximation:

```
clr
smt in th out
eul out g lim in
mul in in in2
mul in2 p88 x
mul in 1.8 y
sub in 0 flg
brp flg 2
sub 0 x x
sub y x sin
dac in ch0
dac sin ch1
end
set th 0.99
set out 1
set dt .0001
set max 100
set g 100
set ch1 1
set p88 .883
set 1.8 1.8758
set 2 2
```



The sine wave is a little cleaner with this method. The third harmonic is down 32dB from the fundamental.