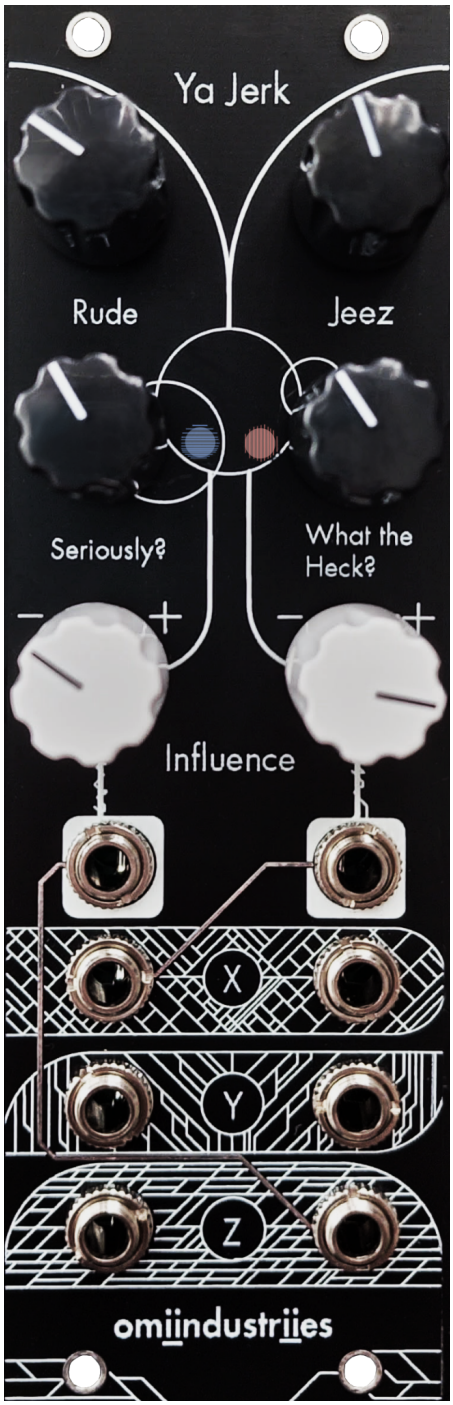


Ya Jerk from omiindustriies is a two-channel chaotic signal generator conceptualized as an argument between two chaotic jerk functions. The two sides each represent a different chaotic entity. It is broken up into two separate but related chaotic signal generators that use JC Sprott's Jerk chaos circuit from the paper "A New Chaotic Jerk Circuit" with some slight variations to optimize it for modular synthesis. Sprott's paper on Jerk chaos can be found here. The two sides use slightly different implementations of the jerk chaos, so you get two sets of related but different chaotic signals. The two channels feature normalization from one side to the other, creating a feedback loop that provides influence over the resulting chaos.

The signals generated from Ya Jerk are not random so much as chaotic, meaning that A: the previous state of the chaotic system influences future states and B: small changes to the system's state lead to large changes down the road. Chaos tends to sit in that sweet spot of unpredictable but bounded where your ear will pick up on patterns as they emerge, but it won't be so repetitive that you can pick out an exact behavior before it happens as the chaos moves through its states never quite the same way twice.

Rude and Jeez control the shape of the outputs; turning clockwise makes the outputs smoother and less jagged. They control how the response of the module at each cycle of the chaos, the kick of the jerk, so to speak. Different settings correspond to the smoothness or jaggedness of the resulting oscillations.

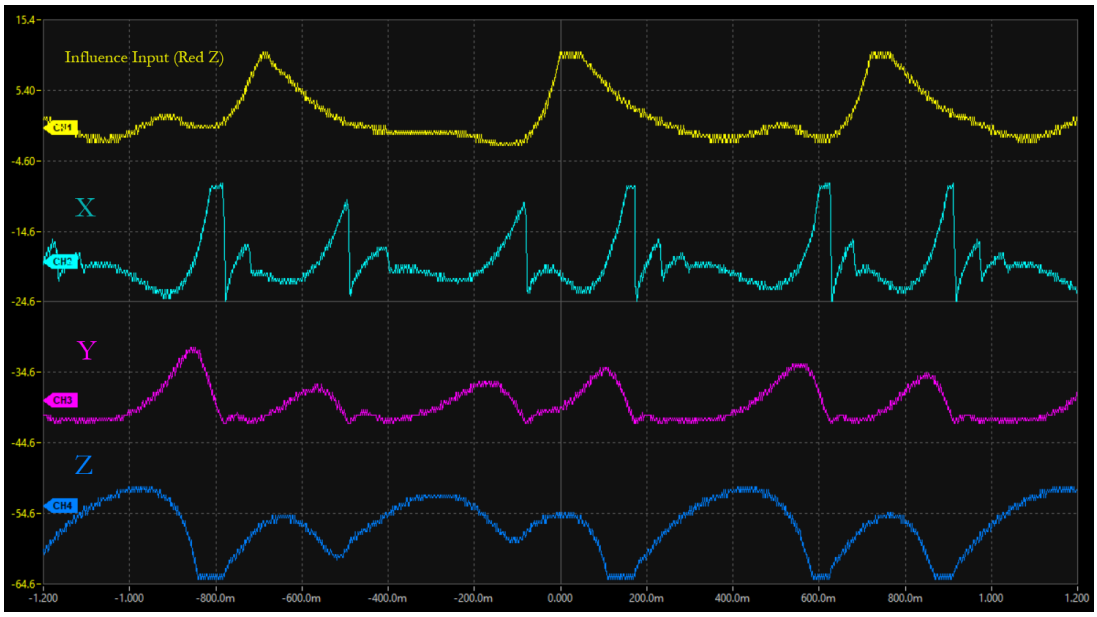
What The Heck? And Seriously? Control the system's relative speed and the amount of decaying bounces. It’s closer to a slew generator than an oscillator frequency control, so turning it clockwise increases the slewing and decreases the speed of the system. Due to the nature of the chaos, the oscillations are often non-periodic, meaning they don’t oscillate the exact same way twice. However, there are periods when the module simply oscillates in a periodic fashion. Finding the sweet spot windows of periodicity and aperiodic behavior is part of the fun of the module. The oscillations are fastest and roughest when the control is turned all the way counterclockwise.

Influence introduces an external signal into the chaotic system. By default, the X output from the Blue side and the Z output from the Red side are normalled into the opposing side, introducing argumentative cross-modulation between the two chaotic systems. But this isn’t a modulation of the frequency, more of an influence on the character of the chaos. Sending gates or triggers into the input causes the module to loosely sync with them, but don’t expect perfectly in-tempo modulation.

The influence controls are attenuverters, meaning in the center, the influence is off; turning clockwise increases the influence positively, while counterclockwise increases the influence negatively. The settings of the influence potentiometers heavily influence the behavior of the chaos. Try setting both attenuverters positively and turning one in the negative direction to see how that affects the outputs.

Each side of Ya Jerk has three outputs; X, Y, and Z. Due to the chaos being implemented slightly differently between the Blue and Red sides, each output is distinct from the others. The different pairs of outputs are similar to each other but have slight variations in their behavior. The outputs are more complex than an LFO but less unpredictable than a smoothed random voltage. The X outputs are the most jagged and jerky, typically behaving most like a bouncing ball with a sharp onset and decaying sinusoidal bounces. The Y outputs are less violent and smoother, and the Z outputs are the smoothest of the three. X and Y are the successive derivatives of Z signal. The outputs swing up to approximately +/-10.5V, but the amplitude of the outputs change with different settings. You may want to use Ya Jerk with an attenuator, attenuverter, or VCA to reign in the amount of modulation.

The outputs from the Blue Channel and Influence Input



The outputs from the Red Channel and Influence Input