RUNG DIVISIONS MANUAL

Rung Divisions pairs a nonlinear feedback shift register* ("nlfsr" based on the core of Rob Hordijk's Benjolin) and a clock divider (CGS Pulse divider) in a small and tactile form factor.

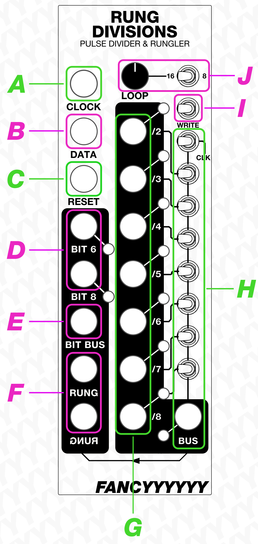
Rung Divisions is designed to be an interactive coupling of linear (clock division) and nonlinear (nlfsr) methods for generating gates, triggers and cv - with means for the two to interfere with each other in interesting and surprising ways. The rungler receives its clock signal from the output of a gate bus which is itself fed from the clock divider.

It will take input from any signal that crosses 0.6V, and as both of the functions are fully analogue (solid state logic, no microcontrollers), they can run at any frequency up to 40kHz.

Rung Divisions has thirteen unique but related chaotic and linear outputs, which are derived from the relationship between three input signals. The outputs come in the form of gates, triggers, variable width pulses and stepped signals. Use them to feed a whole patch with complex rhythms, trigger fills, and cv, or run it at audio rates for a variety of sub oscillations, vocal tract styled pulses, stepped chiptune waveforms and clocked noise.

If you've ever used a benjolin you'll know that this kind of circuit really excels when moving between audio and sub audio rates, with sputters of chaotic rhythms and unpredictable patterns - now you can interface those behaviours with any other equipment or type of signal you can imagine.

*A nonlinear feedback shift register is not really a "thing". But it's a useful concept to distinguish the rungler from a *linear feedback shift register* (often used as a psuedo random noise source). The output of a lfsr is a linear function of its input (often some XOR combination of the data in the register). This is not necessarily the case with the rungler; as there is some *other* signal interfering in the feedback path that may or may not be sufficient for nonlinearity.



A: Clock input for the clock dividers. (See the clock and data input buffer diagram opposite for how the ADC converters work) Note that the clock signal for the rungler is dervied from the divisions of this clock input through the gate bus.

C: Reset input for the clock dividers (use a waveform with a sharp edge). Sets all divider counts to 0 on a rising edge. Patching any of the modules gate outputs here will in turn affect all other outputs downstream resulting in interesting swings and nonlinear patterns.

G: Divider gate outputs (0-8V). Gate width is one cycle of the clocking waveform.

H: Bus switches and gate bus output. Each clock division (and a buffered version of the clock signal) has a switch to send it to the gate bus. The gate bus is a series of "or" combiners. Combining the divisions provides complex rhythmic patterns and can be used as a sub oscillator mixer at audio rate. The bus output is the clock signal for the rungler. With no switches active the rungler will not be clocked.

B: Data input for the rungler. See the clock and data input buffer diagram opposite for how the ADC converters work, and the signal flow diagram (below) for a detailed description of how the final data signal is derived.

D: Individual bit outputs from the rungler. Bit 6 and bit 8 (0-8V). There's a delay of one clock cycle between the gates on these outputs which makes for interesting patterns when driving two voices that "follow" each other. These outputs are variable width gates - the gate width is dependant on the amount of positive data cycling through the rungler.

E: Bit bus output. The rising edges of bits 6, 7 and 8 from the rungler are converted to triggers and combined together. This rapid stream of triggers is usefull for fills and complex rhythms that are higher in frequency than the majority of the other outputs.

F: Rungler outputs. The top output is the rungler cv implemented in the same manner as on the Benjolin (bit 6 of the rungler is the most significant bit). The lower output has the reverse encoding (bit 8 is the most significant bit). Generally the two signals exhibit contrapuntal movement. These signals can be used in a similar manner as you would a two stage analogue shift register (but bears no resemblance from an engineering standpoint). The bit gate outputs pair well with these two cvs as the msb's are tied to the respective gates going high. (See below for more info on digital to analogue converters and most significant bits.)

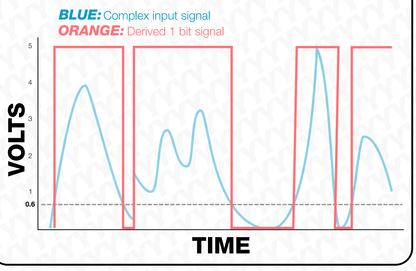
FANCYYYYYYYYYYYYYYYY

I: Write switch. Switch for manually writing data to the rungler (see below for more info on what this means). The write switch overrides any other data input; this means you can manually alter the data in a loop or interfere with the chaos. Push the switch left to force the data input low, pushing it right will force it high. At audio rates the switch will very quickly saturate the rungler with positive or negative values.

J: Loop controls. When the switch is in the left position the potentiometer is active, and sets the chance of new data entering the rungler. When fully ccw all new data will make it in, fully cw the runlger loops a 16 step pattern - many interesting chaotic states lie inbetween the two. With the switch in the right position the rungler loops an 8 step pattern. Note that the potentiometer has no effect on the 8 step pattern or the write switch; the write switch can overwrite the looping data.

CLOCK AND DATA INPUT BUFFERS:

The clock and data inputs each feature a 1 bit Analogue to Digital converter (a comparator) with a threshold of 0.6V. Any signal crossing the threshold will be converted to a square wave, the pulse width of which is dependant on how long the signal stays above the threshold. When the input signal falls below 0.6V the comparator goes low. This means you can use any arbitrary / complex signal to drive the clock divider and data input of the rungler.



RUNGLERS!!! WHAT ARE THE $#%$ING THINGS??

A rungler is made up of two different types of logical blocks; a shift register and an XOR gate. The digital output of these blocks is encoded through a digital to analogue converter into a stepped waveform or "stepped havoc wave".

Some background: A shift register is a type of *sequential* logic (as are the counters that make up the clock dividers on rung divisions). *Sequential* logic can be defined in opposition to *combinational* logic (the gates you know n love: XOR, OR, AND, NAND etc.). The output of a *combinational* logic gate is dependent on the present state of the inputs. *Sequential* logic is affected by both the present state of the inputs, and the history of previous inputs.

A shift register is a kind of primitive ditigal memory / delay, and for our purposes they have two inputs: clock and data. The data present at the input is shifted along a series of memory stages every time the clock signal goes high. The diagram below represents an 8 stage shift register. When the clock goes high, the data input is stored in bit 1 (in this example all the other bits are low).

At the next clock pulse the data is passed to bit 2, so now the data at bit 2 is delayed by two clock periods. The data input is now low, so bit 1 is also low. If we continued this process, then that one bit of data would be passed all the way to bit 8 and be delayed by 8 clock periods.

Schematics for Rung Divisions are available in full at fancysynthesis.net

The data input to the rungler is not as simple as our previous ideal example, and this is where the XOR gate logic block comes in. Bit 8 of the rungler is sent to one input of an "exclusive or" (XOR) gate, the other input of which is the input from the data jack:

It's the combination of the shift register delay, and the nonlinearity in the feedback path that makes the Rungler behave according to the principles of chaos theory when driven / fed back to two oscillators.

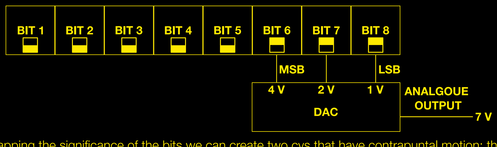
The shift register used in the rungler is an eight stage *serial in parallel out* type shift register, and has three (parallel) outputs that are important to the rungler signal generation at memory stages (bits): six, seven and eight.

These three digital bit outputs are encoded through a digital to analogue converter (DAC), whose output is the stepped rungler signal (this signal is fed back to the driving oscillators of the benjolin to cause chaos). In its original implementation, bit six is the most significant bit in the DAC conversation. Rung Divisions brings an additional encoding to the front panel where bit eight is the most significant bit.

Great... but what's a bit and what does it mean that some of them are more significant than the others?

A digital to analogue converter takes some number of digital bits and converts that information into an analogue signal. In our case we have three bits of binary (0 or 1) data to encode. Put very simply DA conversion weighs the bits and sums them together. In the case of our three bit dac: the most significant bit is weighted to make up 1/1.75 of our total output, the next bit is weighted 1/3.5 of our output, and the least significant bit is 1/7 of our output. Each bit down the chain has half the significance (weight) of the previous bit.

Taking the example below (with voltages chosen to simplify the maths): our total output voltage is seven volts, bit six is the most significant bit and as such is weighted to contribute 1/1.75 of our output signal - which is four volts, and so on. So if only bit six was high our output would be four volts.



By swapping the significance of the bits we can create two cvs that have contrapuntal motion: the first signal is a lower voltage when bit six is low, and vice versa. The output voltages of the rungler DAC are different to those above, but the principle is the same. The total number of possible values our analogue signal can be is 2 to the power of the number of bits x 2: so for us that's 2^6 =8 as this represents the number of zeroes or ones available in our three bits of data.

SIGNAL FLOW