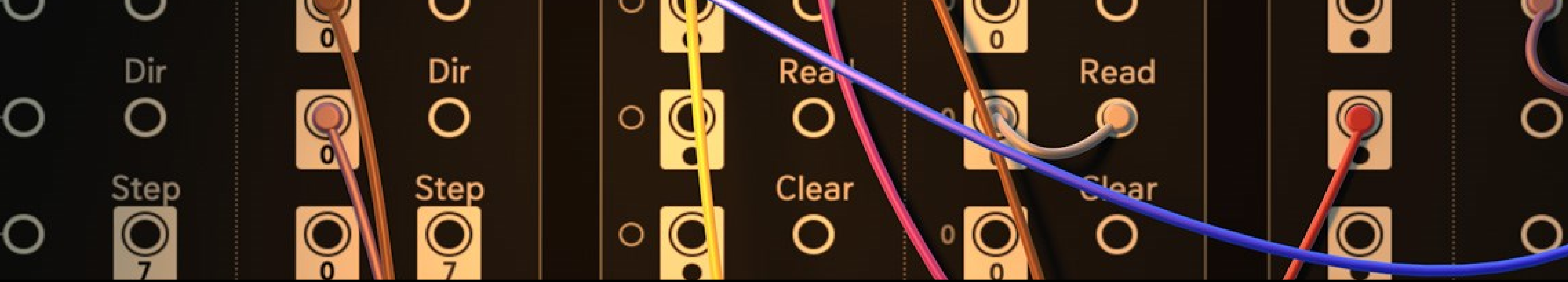
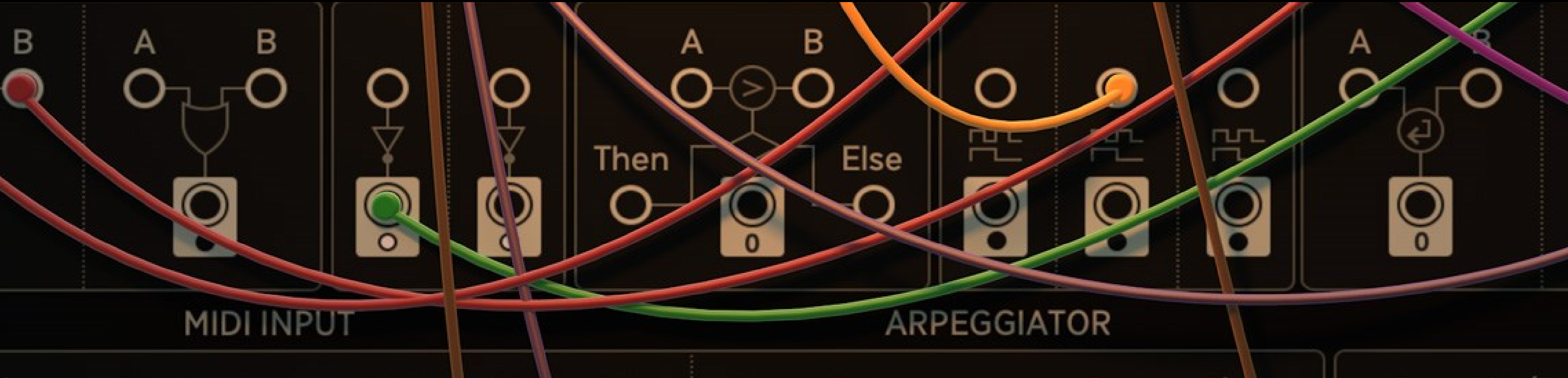


Nest

USER MANUAL



What is Nest?

Nest is what you could call a circuit bending sandbox MIDI generator. It is like having a MIDI Sequencer laid out on the breadboard, having all the internals ready to play with. Nest provides all possible MIDI sequencer functionalities, but on a modular circuit level. Inputs and outputs of imaginary, (yet based on real) ICs can be connected in order to create pitch, gate and modulation signals, and sent to 8 individual MIDI voices. These voices can be assigned to 4 VST2 Plug-ins, internal sound generators and 16 MIDI channels, including automation of Plug-in parameters and MIDI CCs.

Nest is an ode to the classic Transistor Logic ICs that all digital hardware is still made of today. ICs such as the 4015 Shift Register, 4029 Binary Counter, and 4051 Multiplexer. Nest arose from the fact that all these old ICs have musical qualities, and how much fun it can be to play with them on a breadboard. It’s a world of its own beauty and we decided to try and bring it to the user. Only now it’s spiced up with the software power of Presets, Scenes, and Plug-ins.

With Nest, you can build countless music making machines, from cascading chord generators, tricky logic beat machines, round-robin note dispatchers, to ever evolving generative music. You decide how deep you want to go. Nest is a challenge, it’s not easy at first, but the effort pays off. Emotion and Logic work together to create a musical world of Man and Machine. With Nest you can have a new sequencing instrument every day, or a brain for your modular system or studio.

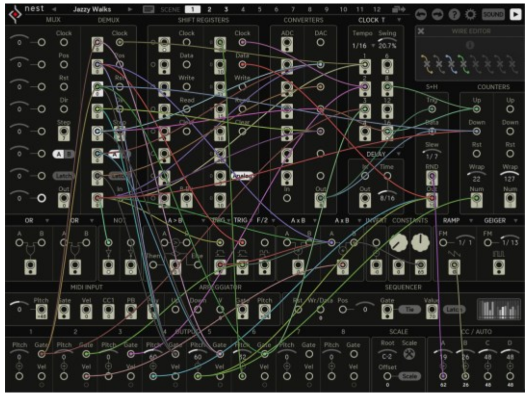
Do you accept the challenge?

Leap into the the unknown

Nest was not developed with specific ‘modules’ in mind, like you would find in a eurorack or 5U modular setup. Rather, it aims to simulate the behavior and possibilities of working directly integrated circuits themselves. In this sense, it has something more in common with circuit bending a polyphonic sequencer than it does with a rack full of filters, oscillators and amps. Instead of routing audio and CV signals, think of Nest as visualizing the midi stream itself – giving you direct access to control and modify note on/offs, pitch, velocity and modulation signals. Your favorite sequencer has been hacked, its internals laid open, and you have an endless supply of wires. The signal flow is now up to you, and the rules are yours to create.

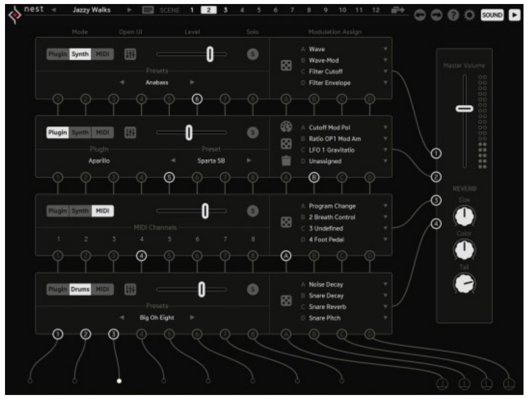
Overview

Similar to many of our products, the user interface is divided into two main pages - the Wire page and Sound page, plus an options panel.
On the Wire Page we create an opus of MIDI data, and on the Sound Page we turn that data into sound.
The math is as follows: We generate 8 voices of MIDI data on the wire page. Then we have 4 MIDI targets on the sound page.
Each MIDI target can decide which of the 8 voices it will listen to. The targets can be set to an internal sound generator, a VST2 Plug-in, or external MIDI out (to control hardware or DAW tracks, for example).



Wire Page

The Wire Page resembles a modular system. Here we interconnect modules to achieve up to 8 voices of MIDI. Presets and scenes are handled on the top of the page. The Wire Editor is shown at the right.
At the bottom, we have the MIDI out section where we trigger voices 1-8, and generate 4 lanes of MIDI CC or Plug-in automation.

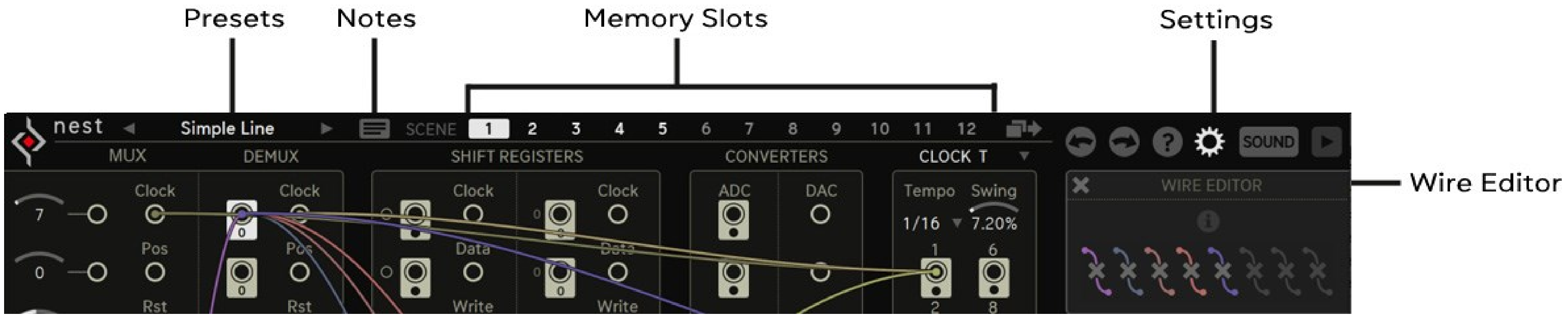


Sound Page

Here we receive the 8 voices and assign them to 4 Sound Channels.
Each of the four Sound Channels offers 3 work modes: Internal Synth, Plug-in and MIDI.
Available are two subtractive synthesizers, one analog resonator and a simple drum machine featuring Kick, Snare and Noise/Cymbal. Each target can load a VST2 Plug-in or send MIDI Out on up to 8 individual MIDI channels. At the output stage we offer a Reverb, which works as a send effect for the 4 internal sound generators.

Get Started

When you open up Nest for the first time and browse through some presets, you may be blown away by the great number of wires and modules, some of which you might not be familiar with. But bear with us, as this is where all the magic begins.



Managing Wires

Wires can be identified by color*. To draw a wire, click and drag out from an input or output port. Drag the wire to an opposite port and release the wire when it snaps in place. When you drag a wire from an In-port, all available Out-ports will be highlighted. Conversely, when you drag a wire from an Out-port, all available In-ports will be highlighted. When a port is selected (by clicking the port directly), all connected ports will be displayed in the Wire Editor.

To delete a wire, you can simply right click anywhere on the wire. For busier projects with many cables you can click on an input/output port and refer to the Wire Editor to view all connections to that port. Click the 'X' on the desired connection to remove that wire. To move a wire, click and drag somewhere on the wire itself anywhere outside of the port it's connected to. Whichever end of the wire you click closest to will be moved out from that port.

*Wire color and saturation settings are available in the Settings panel.

Multiple wires can be stacked to the same In-Port. In the same way, multiple wires can be dragged out from the same Out-Port. If wires are stacked on inputs, their data is added (for example, a wire passing a value of 10, and a wire passing a value of 5 will be added at the In-port for a value of 10+5=15). This saves us lots of module space, as mathematical adding is now included in the main workflow. When it comes for adding / stacking gates, we included 3 gate-to-trigger modules in order to make gates short enough so that they don't overlap each other (i.e. multiple gate sources arriving at the same port).

Presets

We created many beautiful presets which often develop over time and might also be spread along several "scenes". Check out the presets and feel free to replace the internal sounds with VST2 Plug-ins from your collection. Some presets require MIDI notes in order to unfold, as they might be based on the arpeggiator, or record notes to the sequencer. Click the preset name in the Preset Browser of Nest to load global presets and save your own work.

Notes

Some projects may have so much going on that when returning to the preset you've lost track of what's doing what. We've included a Notes panel for you to document how the patch is constructed, in case you need to jog your memory. Enter any critical information into the notes field and it will be saved with your preset.

Design up to 12 Wiring Scenes

The wiring page offers 12 memory slots (switchable by MIDI) for variations of your patch. These 12 scenes not only cover the wiring, but also every module and button state, parameter value, and setting of everything visible on the Wiring Page. Not only that, but the on/off state of the 8 Voice and 4 Modulation assignments on the Sound Page are included in the scenes too. This makes it possible to tailor a Nest performance exactly to your song, as each scene can have its own pitch and scale settings. Build up a drum pattern by adding new voices per scene, transpose a chord progression with different scale offsets per scene, or just completely change all the wiring connections for a new patch. The only limit is your imagination!

Handling scenes is as easy as possible:
Click the source scene.
Click the copy-to button (to the right of memory slot 12), everything starts blinking.
Click the target scene, blinking stops, success!

True to the Sugar Bytes way, the 12 wiring scenes can be instantly switched by MIDI note. Scenes 1-12 correspond to the first octave of MIDI notes (notes 0-11). Scene switching is beat synced to 1/16th notes, so that your patterns always play in time.
For convenience, empty scenes are displayed gray, and scenes containing data (wiring connections) are highlighted white.

Wire Editor

This Module helps with the wiring, especially when things get messy. Click on a port and the wire editor will show all connections of the selected port. You can hover over a wire to read the Ins and Outs, delete a wire with the X below it, or delete all wires with the upper left X. A wire can also be deleted by right-clicking it directly, but there is a certain error-quote, as wires might be overlapping too much in order to right click a certain wire.

Clock Start

Nest has several clock options. Clicking the Gear Icon on top lets you access the 'Clock Source' options. In INTERNAL mode, Nest will run on its internal clock*, only retriggered by your DAW start/stop. If the clock option is set to HOST, Nest will sync to your host's clock. When set to MIDI mode, Nest's sequencer will begin when it receives a MIDI note, for as long as the MIDI note is active.
*To set the internal tempo, refer to Window>Tempo in your computer's menu bar. Or use the shortcut, Command + T.

MIDI Learn / MIDI CC Assignments

Nearly every control in Nest is MIDI assignable.
Right click on any control to open the MIDI Learn popup window.
Click 'learn' to assign an incoming CC to that control.
Click 'clear' to remove the assignment.

When using the Standalone version of Nest, if CC Preset Isolate is enabled in the MIDI Settings, then the MIDI Learn settings won't change when loading new presets. If CC Preset Isolate is unchecked, then the preset's individual CC Map will be recalled with the preset.
Within a DAW Nest's MIDI Learn settings will be saved with the project.

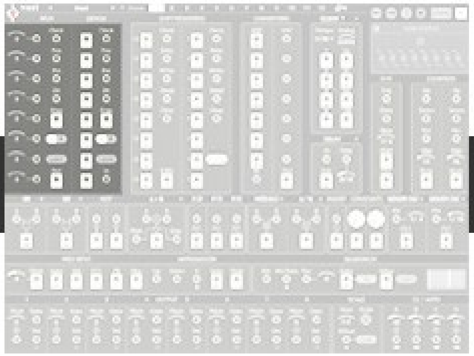
Audio Recorder (standalone)

An additional option Nest provides is an audio recorder (standalone only). With this utility, you can record perfectly looped performances up to 64 bars long, or up to 16 minutes long. This is a great way to record a performance, or make stems and loops to use in your DAW or other samplers. You can access the recorder by going to Window>Audio Recorder, or with the shortcut, Command + P.

Let‘s do the Modules!

In order to use Nest to its fullest potential you need to the tools at hand.
Learning the modules will truly allow you to unleash the beast in the Nest..

Multiplexer & Demultiplexer

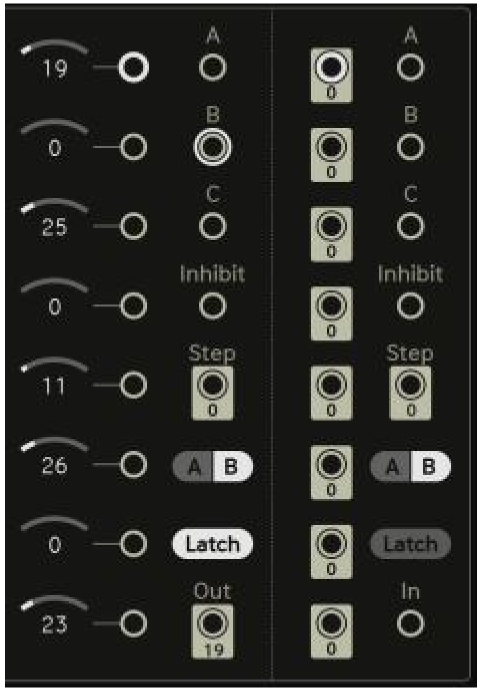


These modules are inspired by the 4051 multiplexer IC. They are basically switches to route 1 input to up to 8 outputs (demux).
Or up to 8 inputs to 1 output (mux). The mux/demux have 2 operational modes - A and B.
Mode A will select the 8 steps sequentially with a clock (this works like an 8-step sequencer).
Mode B will select individual steps with a 3-bit address (i.e., the combination of on/off states routed to the ABC inputs).
This is how the real 4051 IC operates.

This truth table shows the possible combinations to select any of the 8 steps in B mode.
Half the fun in Nest is happy accidents though, so don't hesitate to plug random clocks or oscillators into the address inputs to see what comes out!

Input States				Select Step
Inhibit	A	B	C	
0	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	1	1	0	3
0	0	0	1	4
0	1	0	1	5
0	0	1	1	6
0	1	1	1	7
1	x	x	x	0

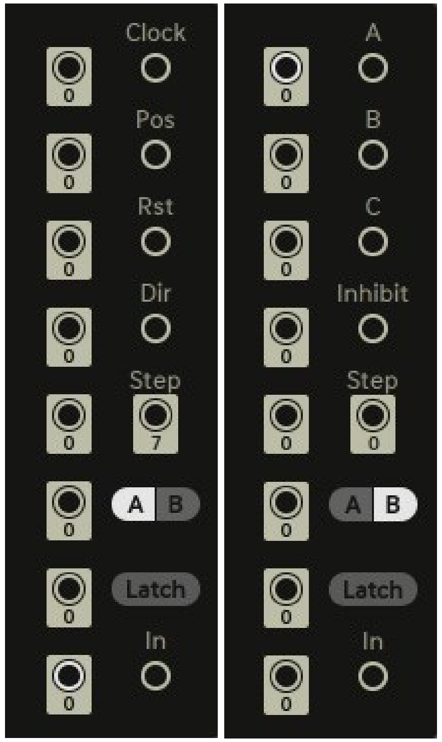
(Note in Nest we count numbers from 0 and up. So, the Mux/Demux counts from 0-7 instead of 1-8).



Multiplexer (MUX)

The MUX is an 8 input / 1 output switch. There are 8 steps which are selected sequentially at the rate of the clock at the clock input. The first thing you may notice are the 8 value sliders running down the left side of the module. These sliders can be adjusted and their value will be sent to the MUX output, when that step is selected. With nothing more than adjusting these values and feeding the clock input with a clock signal you can create a classic 8 step sequencer.

It doesn't stop there. Each step has its own input. You can feed these inputs with any signal - from the shift registers to the oscillators, to different divisions from the clock module. Now, when one of those steps is selected, any data present at that step's input will be forwarded to the MUX main output. This is how you can select data from different modules to be sent to 1 source. Remember the value sliders? Whatever value is set by the slider will be added to the data on that step's input. This is useful to add or subtract from your 8 input signals.



A Mode:
Clock: Send individual triggers to push the MUX one step forward or backward, according to Dir.
Pos: Offset the current step position from 0-7 steps. Values wrap around 7.
Rst: Restart the internal counter (restarts the MUX to first step)
Dir: Direction that the MUX reads its inputs. 0 = forward/up, 1 = backward/down.

B Mode: The combination of address inputs ABC will select different MUX steps. (See truth table for bit addresses)
A: Selects step 1.
B: Selects step 2.
C: Selects step 4.
Inhibit: Address operation is put to rest. Freezes MUX on step 0.

Inputs: Step value sliders (x8), MUX inputs (x8)
Output1 (Step): Outputs number values from 0-7 depending on current step.
Output2 (Main): Multiplexer output. Outputs the value present at the current step's input (unless Latch is active)
Latch: Sample the output value with the clock, to maintain a single output value for the lifetime of a step.

Demultiplexer (DEMUX)

This is the same as the 4051 MUX, but the other way around. It is a 1 input / 8 output switch. There are still 8 steps which are sequentially selected at the rate of the clock, however this time, it is 1 input signal which gets forwarded to 8 individual outputs. You could dispatch a rhythm or melody over 8 different voices, or direct a step sequence to a spread of voices, one at a time.
Operationally it is no different than the MUX, having the same bit addresses in B mode, and the same clock and counter functions in A mode.

A mode：

Clock：Send individual triggers to push the DEMUX one step forward or backward, according to Dir.

Pos：Offset the current step position from 0-7 steps. Values wrap around 7.

Rst：Restart the internal counter (restarts the DEMUX to first step)

Dir：Counter/Sequencer direction. 0 = forward/up, 1 = backward/down.

Output (Step)：Outputs number values from 0-7 depending on the current step.

B Mode：The combination of address inputs ABC will select different DEMUX steps. (See truth table for bit addresses)

A：Selects step 1.

B：Selects step 2.

C：Selects step 4.

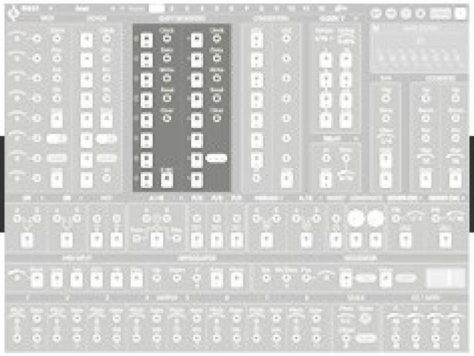
Inhibit：Address operation is put to rest. Freezes DEMUX on step 0.

Input：Main input. Forwards the input data to the current step's individual output (unless Latch is active)

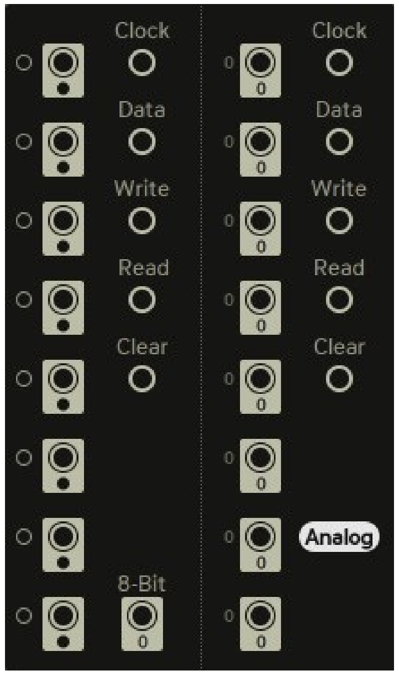
Outputs (x8)：The input data is sent to these outputs, which are selected by the clock+position+direction in A mode, or the address in B mode. The output value will be present for the lifetime of a step unless Latch is active.

Latch：Input value is written to the current step when the clock is triggered, until the step is selected again, turning the DEMUX into an 8-entry memory cell.

Note: All Nest inputs which expect a logic '0' or logic 1'' will simply treat values greater than 1 as 1. So, the clock input advancing the MUX or Shift Register, when receiving a '5' will just treat it as a 1. In this way, many signals can be used at logical inputs.



Shift Register

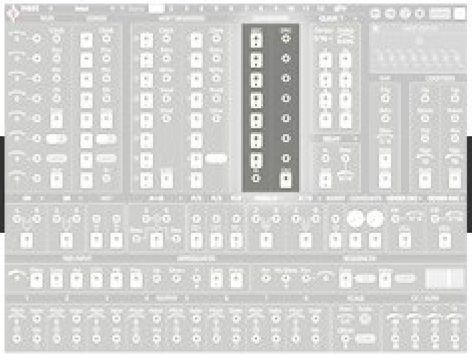


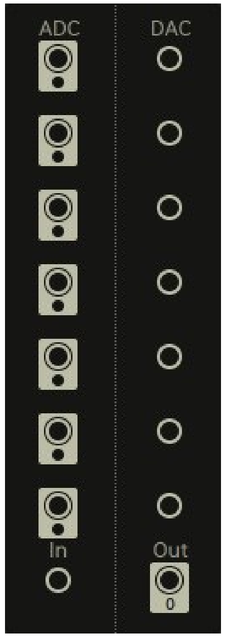
Derived from the classic 4015, the shift register is a data recording device. They allow you store number values within their eight 'memory cells'. What makes the shift register unique is that the cells are arranged in a line, with the output of one cell connected to the input of the next cell. This allows the data from one cell to be 'handed over' to the next cell, until the data is finally discarded (or fed back to the input to cycle continuously).
The thing about the about the shift register though, is that it has 8 simultaneous outputs. This allows a melody, for example, to be passed into the data input, and that melody can be split out among the 8 outputs.

Nest has 2 shift registers, one of which can be switched to an analog shift register (ASR). The analog mode is generally useful for processing and generating pitch, velocity or modulation signals. In 'digital' mode, both shift registers are useful for generating gates, triggers, or clocks.
Both shift registers have a 1 preset memory which can be saved and accessed by the write and read inputs.
A good image for a shift register is a bucket brigade. There are 8 individuals, each holding a bucket. The clock tells them to pass their bucket to the next individual in line. When the clock arrives, person #1 will pass their bucket to person #2, while person #2 passes their bucket to person #3, and so on. In the case of the digital shift register, the contents of the bucket represent a logic state (0 or 1 - full or empty). In the case of the analog shift register, the buckets may contain any number value from 0-127.

Clock:	When the SR receives a clock (logic 1), it will sample data into the first cell, and shift all cell's data over by one position.
Data:	Data at this input will be sampled into the shift register on the clock pulse.
Write:	Sending a logic 1 to the *write* input will save all cell's current values to memory. The memory value is seen to the left of the cell.
Read:	Sending a logic 1 to the *read* input will recall the written values to their respective cells.
Clear:	Trigger this to erase all data from the shift register.
Data out (x8):	8 individual outputs of the shift register's cells.
8-bit Output:	Outputs a value from 0-127 which is equal to the added bits in the shift register.
ANALOG switch:	You can switch SR2 to analog mode in order to process values from 0-127 (for pitch, velocity or modulation.)

Converters





Here we have a 7-bit AD/DA converter. The use of such a converter is to turn analog voltage into digital numbers (bits) and vice versa. A bit is simply a 0 or a 1. In short, the converter will take 'analog' values from 0-127, and spit out a unique combination of seven 0s or 1s (ADC). Or conversely, it will take the sum of seven digital inputs, and output a unique 'analog' value of 0-127 (DAC). This is a perfect range for working with MIDI.

Key to understanding the converters is to know what numbers the Bit input and outputs represent.
They are multiples of each other. The converter values of 1-7 are as follows: 1 + 2 + 4 + 8 + 16 +32 + 64. Or:

AD/DA 1 = 1
AD/DA 2 = 2
AD/DA 3 = 4
AD/DA 4 = 8
AD/DA 5 = 16
AD/DA 6 = 32
AD/DA 7 = 64

How to use this will become clear below.

DA Converter (DAC)

The DA converter has seven inputs and a single output. Each input accepts a 'bit', that is, a 0 or 1. These can come from clock signals, square LFOs, MIDI gate input, and more. Now, because we already know the values each input represents, let's try making some numbers:

If we take the first two inputs (DAC1 and DAC2), and we connect a '1' to both, the output sum will be 1 + 2 = **3.**
If we connect the first 3 DAC inputs, because we know that they represent 1 + 2 + 4, then our output sum will = **7**.

Now, let’s work from the opposite direction. Let’s say we want the number ‘42’. It’s simply a matter of finding and adding the appropriate inputs.
DAC6 = 32, DAC4 = 8, and DAC2 = 2. Send a ‘1’ to each of those DAC inputs, and then voila, 32 + 8 + 2 = **42**.

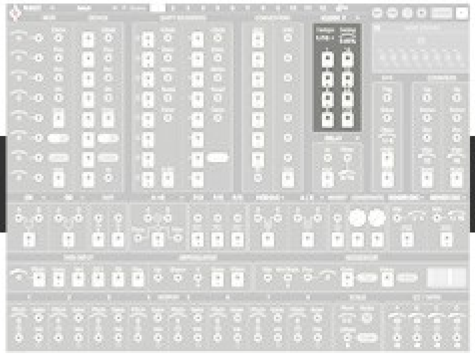
How to use this musically? By sending different triggers and gates to the DAC inputs, you can create repeating number patterns at the output.
This pattern can be used for pitch, velocity or modulation. It can be used as a unique arpeggiator or sequencer, or used to transpose Nest’s voices.
Now that you know the theory behind the DAC’s operation, you can be deliberate, or you can let happy accidents run wild.

AD Converter (ADC)

The AD Converter has one input, and seven individual outputs. The input accepts ‘analog’ values from 0-127, and the seven outputs spit out ‘bits’, or 0s and 1s. The ADC is the exact opposite of the DAC. The outputs all represent the same integers, which are provided above.
Let’s see how it works:

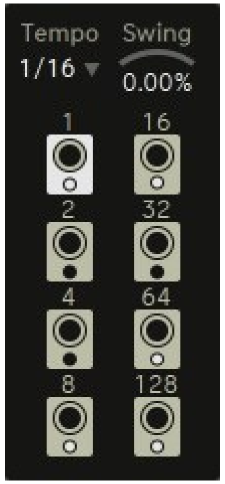
Using our previous examples:
When a value of ‘3’ is presented to the input, then ADC1 and ADC2 will light up. That’s because the value of ADC1 = 1 and the value of ADC2 = 2.
When they are added they = **3**. The same as our input value.
When we present the input with a ‘7’ then ADC1, 2, 3 will light up. We know that is because they represent 1 + 2 + 4 = **7**.
I’m sure you can guess by now that the value of ‘42’ will cause ADC2,4,6 to light up.

But the question remains, how can we use the ADC musically? Well, each time an output lights up, a ‘1’ is spit out. That ‘1’ can be used as a clock, a gate, or a trigger. It could be multiplied with the Math module to transpose individual MIDI voices. You could send a melodic step sequence to the ADC’s input, and for each note, a unique 7-bit output combination will be generated. You can use this to trigger drum voices, or additional melodic voices. Because the ADC outputs have a direct correlation to your input melody, the output rhythm is not random. With seven trigger/gate outputs there are many possibilities.



Clock

Here is a handy clock module with 8 division outputs.
There are many uses for the clock such as triggering voices, advancing shift registers or the MUX/DEMUX, or generating numbers with the DAC.



Three clocks are available：

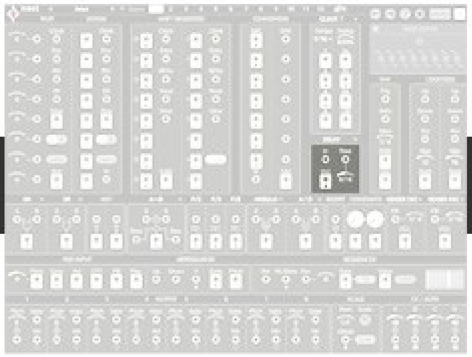
Straight (Clock)：8 Straight clock divisions, based on the global division.

Triplet (Clock T)：8 Clock divisions including triplets.

Binary (Binary Counter)：8 clocks in form of a Binary Counter. Syncopates a bit, has the feel of an inverted clock.

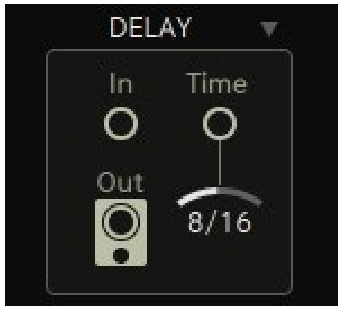
Tempo：Global Tempo division, which is then divided down further by the clock divider.
Swing：Every 2nd step of the global tempo divider is slightly delayed, creating a classic swing feel.

Tip：Use the binary counter to accurately count numbers in the DAC, and in the MUX/DEMUX B mode.



Delay/Hold/ClockDivider/Envelope

Here we have 4 modules to choose from.



Delay

The delay module can delay any signal by 1 to 16 sixteenth notes.
Works just as well on pitch and modulation as it does on triggers.

In：	Data input for signals to be delayed.
Time：	The delay time, given by the number of 16th notes. Has an input for modulation of delay time.
Out：	Output of the delayed data.

Hold

The hold module can lengthen or shorten a gate/trigger. This can be useful for gates whose durations are too long, or for triggers whose durations are too brief.

In：	Data input for the thing that must hold on.
Time：	Hold time in 16th notes.
Out：	Data output for the stuff that holds on.

Clock divider

Sometimes a clock is not slow enough, a rhythm is too busy, or you want an odd clock but only have even clocks. This is when the additional clock divider is our friend.

In：	Data input for the clock or trigger signal.
Time：	Division factor, where 3/16 means that the incoming clock is divided by 3.
Out：	Data output of the divided clock.

Envelope

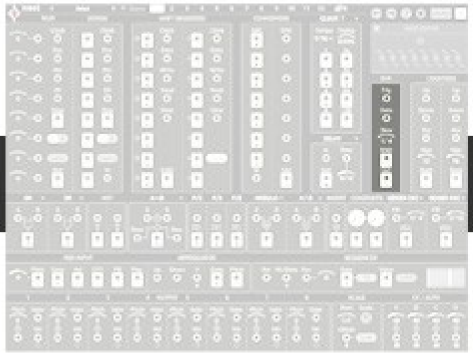
The envelope generates a falling curve, like a ramp. When the envelope is triggered it will ramp from its highest value, down to 0. It will decay for the duration set by the 'time' setting. This can be useful for parameters you want to gradually or quickly decay downward – perhaps the wet level of a reverb, or the velocity of a busy hi-hat pattern.

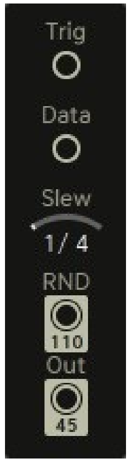
Be aware that the envelope outputs a value between 1 and 0, and must be multiplied up for greater values. Refer to the Math module for multiplication.

In: Trigger input for the envelope.
Time: Decay time given in 16th notes. Max decay time is 64/16
Output: Envelope output. Be default the envelope outputs a value between 1 and 0. Multiply the envelope output for greater values.

Tip: Several signals require multiplication to unlock their full range. These include the Envelope, and the Sine/Ramp/Saw/Noise shapes of the Oscillators. Use the AxB Math module and a Constant value for precise multiplication. Alternately, drag out multiple wires from an output, and stack them on a single input. This has the same outcome as multiplication, where the number of stacked wires = your multiplier value. This is useful for multiplying data by smaller values, or when you run out of available Math modules.

S+H



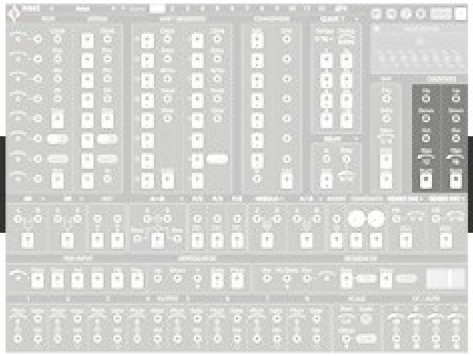


The uses for sample and hold are many, but it's operation can be described simply: a snapshot of data is saved every time the S+H receives a trigger. It's like taking photograph of a live data stream. The only thing to overpower the S+H is the Shift Register, which packs 8 sample and holds in one module.

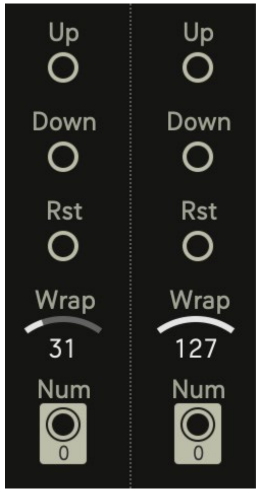
Operationally, how it works is this:
The S+H has two inputs, 1) a trigger input, and 2) a data input. When the S+H receives a trigger, the current value at the data input is sampled, and 'held' to memory. This sampled value is then available at the output. It will be held indefinitely, until a new trigger is sent to the S+H, and a new value at the data input is sampled.

Sample and hold was often used as a 'random LFO' in old synthesizers, but it can do far more than just that. For example, a fun patch is to create counter melodies and bass lines by sampling parts of a pitch sequence with different triggers.

Trig:	When this input is triggered, the data input (and the internal randomizer) will be sampled.
Data:	Input data here to be sampled.
Slew:	Slew will smooth the transition of the output data, when the data changes values. The slew (or transition) rate is determined by divisions of one bar. The maximum setting produces hard transitions, and the minimum produces entirely smoothed changes.
RND:	The S+H has an internal randomizer that does not require external data. A random value will be generated when the trig input is triggered.
Out:	The sampled value is output here, until a new value is sampled with the trig input.



Counters



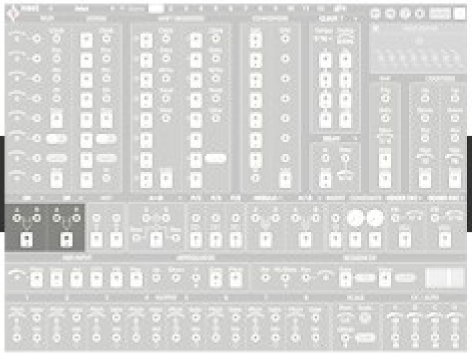
These counters increment values up to, or down from a specified number, set by *Wrap*. A clock/trigger event at the input will tell the counter how much to count by. So, a clock with the value of ‘1’ will tell the counter to count up by ones: 1, 2, 3, 4, 5, 6, etc. And a clock with a value of ‘2’ will tell the counter to count by twos: 2, 4, 6, 8, 10, etc. This occurs until the counter arrives at the *wrap* number, at which point it starts from 0 again.
The counters have two inputs: *Up*, and *Down*. Feeding the *Up* input with triggers will cause the counter to count up. Feeding the *Down* input with triggers will cause it to count down. Feeding both *Up* + *Down* simultaneously will cause the counter to count in both directions. Imagine walking a staircase, taking 5 steps up, then 2 steps down, and repeating.

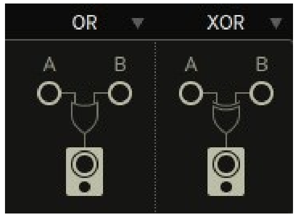
Because the counters can cover a broad range of values, they are great for creating sequences, arpeggios, and modulation. They are also indispensable utilities for a number of operations, especially when used in conjunction with the Comparators, the If/Else module, and any *Pos* (position) input.

Up:	Trigger an up-count. The up-count has the value of the trigger. So, if you send a 3 here, the result is incremented up by 3.
Down:	Trigger a down-count. The down-count has the value of the trigger. So, if you send a 3 here, the result is decremented by 3.
Rst:	Resets the counter to 0. As with all *Rst* inputs in Nest, they are reset when the host clock restarts.
Wrap:	The counter will wrap around at this number. It is the maximum value, which will keep results below this number.
Num:	Data Output of the counted numbers.

Tip: A clock signal is a signal that oscillates between 0 and 1. A clocked event usually occurs on the ‘rising edge’, or when the signal moves to 1 from 0. To create clocks that oscillate between 0 and a higher number, use the AxB Math module to multiply the clock to a higher value. Or, just drag out multiple cables from the same clock, and stack them at a single input.

Logic



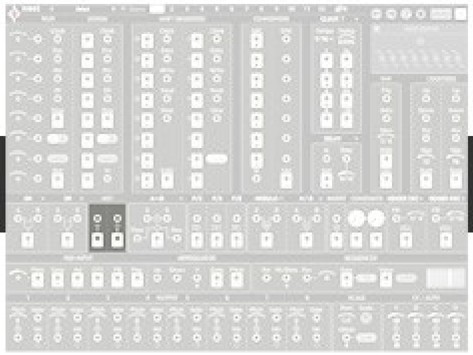


Logic Gates are the key operators of digital computing. They operate with binary values, 0 being LOW (no signal), 1 and above being HIGH (signal). Logic Gates force their inputs follow specific rules, such as: ‘if two triggers happen at the same time, don’t output a trigger’ (NAND). Or, ‘only output a trigger if both inputs happen at the same time’ (AND). For convenience, we’ve included Comparators (>, <, =) which work with all number values (but output a logical 0 or 1).

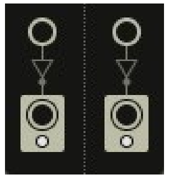
Many interesting rhythms can be created with logic gates. Use them to merge triggers, or create new rhythmic sequences from existing ones. Experiment by plugging different triggers and clocks into the *A* and *B* inputs, and select a rule till you find something you like.

OR	Output will be HIGH if one or both inputs are HIGH
NOR	Output will be LOW if one or both inputs are HIGH
XOR	Output will be HIGH if one input is LOW and the other is HIGH
XNOR	Output will be LOW if one input is LOW and the other is HIGH
AND	Output will be HIGH if both inputs are HIGH
NAND	Output will be LOW if both inputs are HIGH
>	Output will be HIGH if *A* is greater than *B*
<	Output will be HIGH if *A* is smaller than *B*
=	Output will be HIGH if both inputs are equal
!=	Output will be HIGH if both inputs are not equal

Tip: The different logical combinations and their outcomes can be found in charts called ‘truth tables’. Please check the Appendix at the end of this manual to view the truth tables for all the logic gates and comparators listed above.

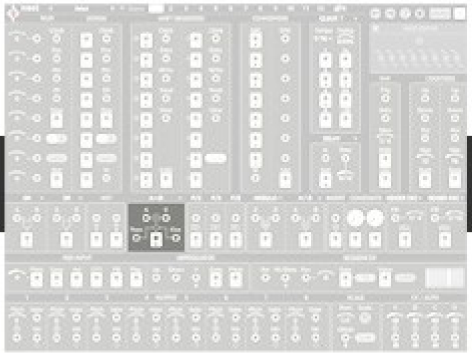


Not

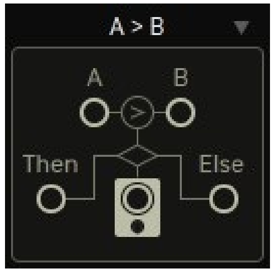


NOT inverts a logic state (HIGH or LOW).
Input values above 0 will come out as 0, while an input value of 0 will come out as 1.

If/Else



The basics of computing come in quite handy, so we included them in the form of an IF/ELSE module.

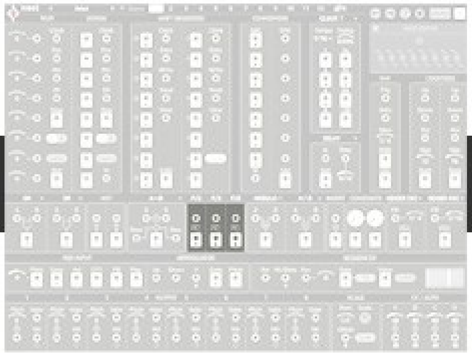


It operates on various conditions, **comparing** inputs *A* and *B*, and **forwarding** inputs *C* (true) or *D* (false) to the *output*.

For example, we can make the statement:
If A is less than B, then we output C, otherwise D is output. That is the (A<B) condition.

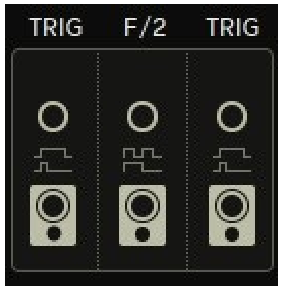
A=B	If input A is equal to Input B, then input C is selected, otherwise input D is selected.
A!=B	If input A is unequal to Input B, then input C is selected, otherwise input D is selected.
A>B	If input A is greater than Input B, then input C is selected, otherwise input D is selected.
A>=B	If input A is greater than or equal to Input B, then input C is selected, otherwise input D is selected.
A<B	If input A is smaller than Input B, then input C is selected, otherwise input D is selected.
A<=B	If input A is smaller than or equal to Input B, then input C is selected, otherwise input D is selected.

Does this all sound too hypothetical? A musically relevant example could be, “When the velocity of my MIDI Input (A) is greater than the threshold of 48 (B), then my kick drum gets triggered by the MIDI Gate (C), otherwise the kick drum is triggered by the Step Sequencer (D). This would enable a scenario where the Step Sequencer is driving a kick drum, until external MIDI notes with velocities above a certain threshold are input, in which case the kick drum is driven from external MIDI instead.



Gate To Trigger and Flop (TRIG and F/2)

Gate To Trigger



A gate has 2 states: OFF (or low/0) and ON (or high/1). A trigger is just a very short gate, usually between 2-5ms. Because gates and triggers only have two states (low and high), when a high state is added to an existing high state, it doesn't produce an effect. It will either extend the duration of the existing high state, or simply be swallowed up by a longer high state. The Gate to Trigger Converter (TRIG) will take a **gate** of any length, and turn it into a short **trigger**.

With this, now you can take gates from two different clocks (like the Giger Oscillator and an 8/16 clock signal) without having the overlapping gate events cancel each other out. An example:

Let's say you use the very slow 16/1 clock, which has a gate ON time of 8/1 (a half bar at 16th note resolution). You use this 16/1 clock to trigger a MIDI voice at its gate input. Now, for the lifetime of that event (which is a half bar), no other gates at that input will be accepted. Send the 16/1 clock to the *Gate to Trig* first. Now, that long gate will turn into a shorter trigger, and will no longer interfere with other gate events at the destination gate port.

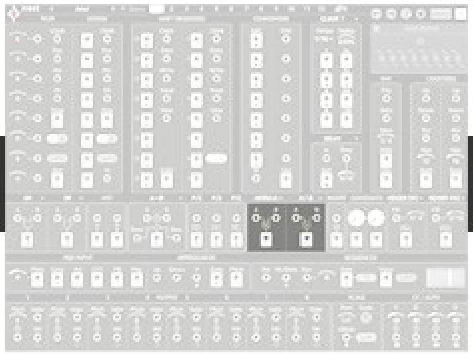
Tip: Sometimes swallowing up shorter gates with a long gate can be advantageous. If you have too many gates going on, or want to add rests to a busy pattern, you can stack a slow gate at the same In-port in order to mute all gates for some time. As long as the input is high, no other gates will be accepted.

If you run out of Gate to Trig modules, use the Hold from the delay/hold/clock divider/envelope module to shorten a gate event.

Flop

Or better known as flip-flop.
It is basically a simple clock divider. An input gate will turn the flop high, and the next input gate will turn the flop low.
Note that the flip-flop does not care about input gate lengths. The input is simply responsible for turning the Flop on and off.

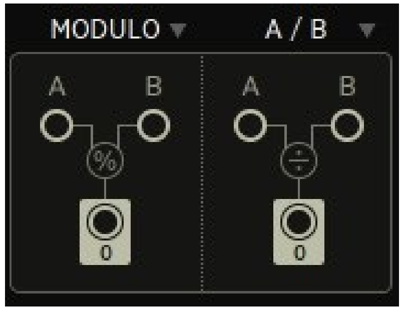
Math



Simple equations for whenever math is needed. This can be for when you need to multiply up the value of the oscillator waveshapes or the envelope signal. Or if you want to subtract 2 octaves (24) from a sample and hold signal.

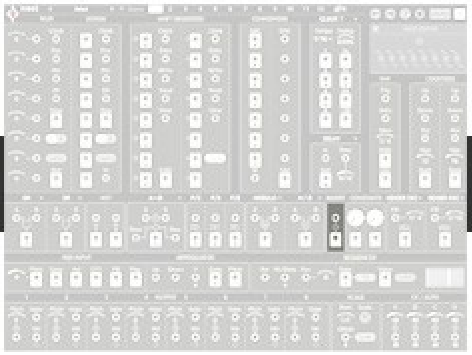
A-B	Input A is subtracted from input B.
AxB	Input A multiplied by Input B.
A/B	Input A is divided by Input B.
Wrap	Input A will wrap around at the value of Input B (never exceeding the value of input B).
MIN	Inputs are compared, the smaller value is forwarded to the output.
MAX	Inputs are compared, the greater value is forwarded to the output.
Average	Standard mean average. Inputs A and B are added, and their sum divided by 2.
Percent	Scale A by B%. (100 at the B input represents a 1:1 scale. 50 at the B input represents a 1:2 scale, etc. (If you want ‘key-tracking’ this is your module.)

Tip: Remember that addition (A+B) is supported by simply stacking wires, therefore it is not included as a math module.



A+B	No MATH module required, just stack wires in order to add their values.
A-B	Input A is subtracted from input B.
AxB	Input A times Input B (Classical VCA).
A/B	Input A is divided by Input B.
MODULO	Input a will wrap around at the value of Input B (never exceeding the value of input B).
MIN	Inputs are compared, the smaller value is forwarded to the output.
MAX	Inputs are compared, the greater value is forwarded to the output.

Invert

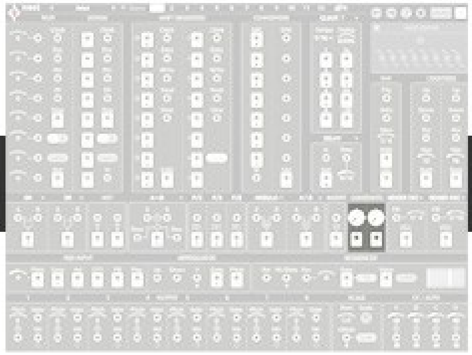


Values get inverted here. So, 1 becomes -1, and 84 becomes -84.
This can be used to make negative numbers, which when added to positive numbers, results in subtraction.
Also, it can be used to make rising values fall, and make falling values rise. A classic example of sidechaining.

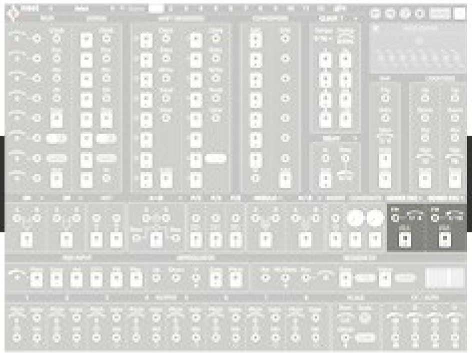
Take these examples: the envelope module outputs a decay shape. When inverted here, it turns into an ‘attack’ shape.
Also, the oscillators can be phase inverted by 180°, creating a mirror image which can be useful for many kinds of modulation scenarios (stereo panning, cutoff frequency of 2 synths, etc).

Tip: The negative values often generated by the Invert module are outside of the MIDI spec. But you can work with it by adding the negative value to higher values above 0. Such as (-12 + 36 = 24.) Just make sure your target value has a suitable positive bias and you'll be well within MIDI ranges.

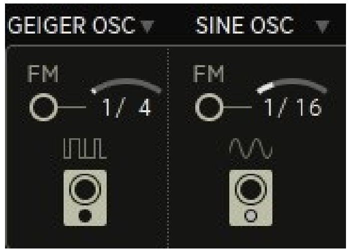
Constants



The constants, simply put, are manual knobs that output a value from 0 to 127. These constants are often used with the MATH modules, and are also useful for setting offsets. Like every other control on the Wire page, they are MIDI automatable, and their values are saved in Scenes.
When getting started with Nest, these Constants can be of great help to determine how all the different modules and inputs operate.



Oscillators

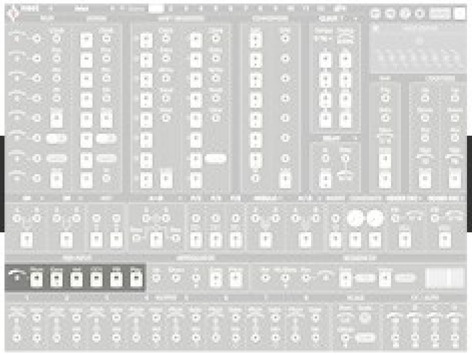


These oscillators are similar to LFOs – low frequency oscillators.
They run at divisions of the host tempo, from **1/1** (one bar) to **1/64** (sixty-fourth notes)
They produce no audio, but provide control signals.
Their output values range from 0-1, and need to be multiplied up to the desired operating range.
Refer to the MATH module for multiplication.
A variety of waveforms are provided, with an input for frequency modulation.

RAMP	Rising ramp
SAW	Falling saw
SINE	Sinus waveform
NOISE	Random
PULSE	Binary waveform (Low/High) with 50% duty cycle.
GEIGER	Binary waveform (Low/High) with random occurrence.

Tempo: Divides one bar by the given amount, to set the tempo of the wave cycle.
FM: Adds the input value to the current tempo setting, to modify to oscillator frequency/tempo.

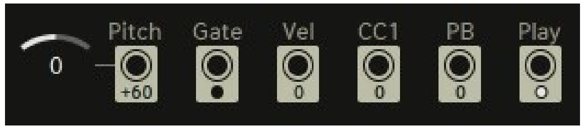
Tip: Dragging multiple wires from the Oscillator output, and stacking them all at the same In-port is an easy way to add/multiply up by smaller values.
E.g.: Drag 5 wires out from the Noise wave, and stack them at an In-port, and you now have tempo synced random values between 0-5.
A great way to save yourself a MATH module for small operations.



External MIDI in

This is the spot where you can route external MIDI into Nest.
These might be used the old-fashioned way, forwarding pitch from your keys to different sound engines.
But as 6 different number sources coming from the outside, it can also bring a greater sense of interaction between Nest and your DAW, or favorite hardware.

Write a beautiful melody into the analog shift register with the pitch output, transpose your entire project with different MIDI notes, or create interesting rhythms in the ADC using MIDI velocity. There are many possibilities before we even consider drawing automation lanes in the DAW.

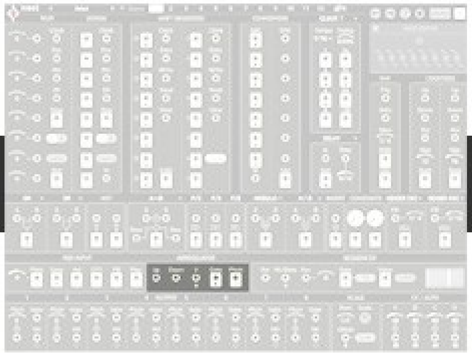


Pitch:	MIDI Pitch of the last note played / entered. Has a Transpose offset slider to transpose incoming MIDI.
Gate:	Incoming MIDI Gate. Signal remains HIGH as long as the note is active.
Velocity:	Velocity value of the last MIDI note played / entered.
CC1:	Mod Wheel comes in on CC1.
PB:	Pitch Bend comes in from -64 to 0 to +64.
Play:	This output stays High as long as the transport is active.

Note that this module outputs MIDI notes 13 and up. Notes 0-12 are reserved for Scene Changes.

Tip: Aren't using the MIDI Input, or don't plan on using it for a current preset? Use the Pitch Output's Transpose Slider as a manual offset source. Providing both negative and positive numbers - great for both addition and subtraction!

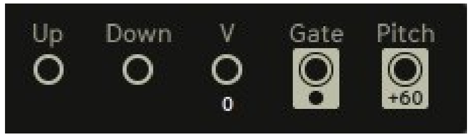
Arpeggiator



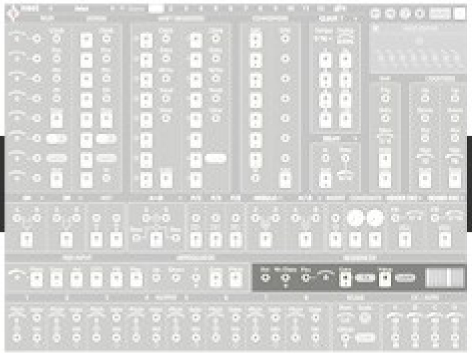
Arpeggiators need no introduction, but the one in Nest is special.
Covering up to 8 voices, this arp enters notes into the voices with an up/down counter, which makes it easy to select voices by triggering the UP or DOWN inputs. The voice stack will be written to memory, when the number of incoming midi-voices rises.

After all notes have been released, a fresh note stack can be written.
To address the voices, the arp uses an Up/Down counter, and a special Voice selector.

Incredible arps can be designed by inputting triggers of various time divisions into the counters.
The Voice Stack can be sorted or unsorted, please select the desired mode in the Settings dialog.

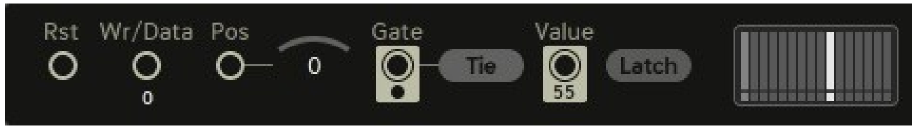


Up: Apply triggers to count up through the input voices. Triggers increment according to their number value (count up by 1s or 2s, etc.)
Down: Apply triggers to count down through the input voices. Triggers decrement according to their number value (count down by 2s or 3s, etc.)
V: Select a voice directly with numbers from 1-8. When a voice is selected, then the counter will count up or down from that number.
E.g., input a ‘4’ and the counter will count up from 4. Also note that numbers behave as triggers at this input.
Gate: All incoming Up/Down/V triggers are merged and sent to this Gate output.
Pitch: The selected voice’s pitch is sent to this Pitch output.



Sequencer

Here we have a full blown 16 step Gate + Value sequencer.
With built in clock divisions, variable sequence length, and control inputs for restart and step position, this becomes a powerful tool to spice up your projects.

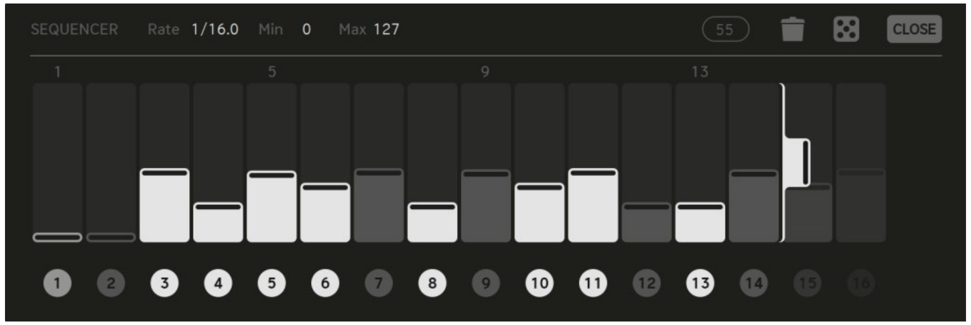


Rst: When a trigger is received here, the step sequencer will restart from the position set by *Pos.*
Wr/Data: Write data into the sequencer with this input. Values will only be written upon modulation to the *Pos* input, or its connected slider.
Pos: Offsets the position of the sequencer's 'playhead'. When the *Pos* value changes, any value present when a wire is connected to the *WR/Data* input will be written to that step position.Values wrap at 15.
Gate: Gate output.
Gate Tie: When this is active, successive gates are tied together, forming longer notes.
Value: Value output (note the Min/Max range in the sequencer window).
Latch: When this is active, only the values of enabled steps will be output from *Value*.

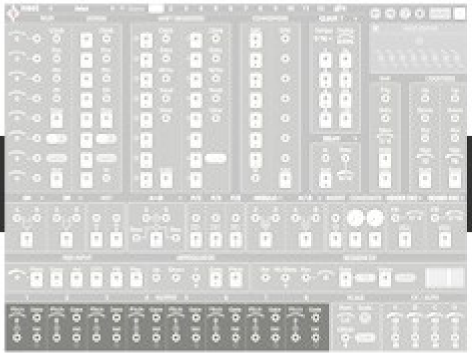
Tip: Remember that gates and values in the sequencer are generally decoupled. An inactive step will not output a gate, but it will still output a value. Turn on Latch to couple active gates and values.

Sequencer Edit Window

Click on the sequencer to open a floating Sequencer Edit Window.
Clicking on a step will activate the gate for this step. Active steps are bright, and inactive steps are dark. Values can be adjusted by dragging up/down. Dragging the white brace marker will reduce the number of steps, the shortest sequence length being 2 steps. When the transport is on, the sequencer will always be running.

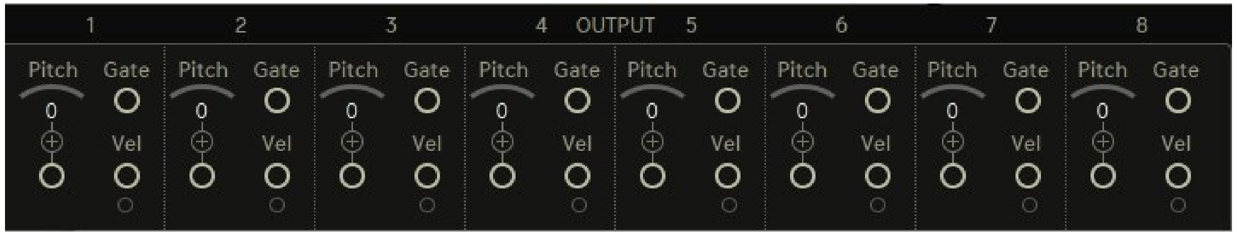


Rate: Clock divider for the sequencer clock. (Independent from the main clock module settings).
Min/Max: Step Sequencer value range.
Value Display: Shows value of the selected step.
Trash Can: Cleans out the sequencer. Click once to activate all steps at max value. Click again to deactivate all steps at min value.
Random: Roll the dice for a random sequence. Click undo/redo if things go wrong.
Loop Length: Sets the number of steps in the sequencer.



Gate/Velocity/Pitch MIDI out

Here is where the magic turns into reality. This is where the numbers, triggers and gates in Nest get converted into MIDI notes, and sent out to the Sound Page. Nest tries to make this as easy as possible. When a gate is sent to the gate input of these modules, a MIDI note is automatically created. If no wires are connected to the pitch input, then the value set by the pitch slider is used. If no wires are connected to the velocity input, then max velocity will be used. Note, without a connection to the Gate input, no MIDI notes will be generated.



Pitch: The input value at the time of a gate on event is the MIDI pitch of that note.
A pitch slider is provided to transpose and offset the voice.

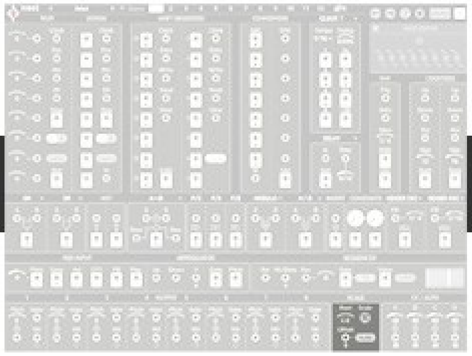
Gate: Low/0 and High/1 values are expected here. Gates must go low before a new note can be created.
The moment a gate goes High, a MIDI note is created with the pitch and velocity values present at the moment of the gate.

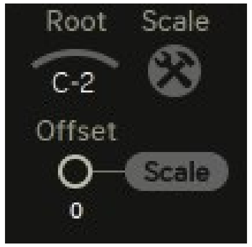
Velocity: Velocity sets the initial amplitude of the MIDI note (among other user definable assignments) at the moment it is triggered.
With no data present here, max velocity is used instead.

Tip: Value changes to pitch and velocity are not recognized after the initial note on/gate on. Pitch and velocity are assigned at the Gate On event only, as per the MIDI spec.

Tip #2: Multiple wires to the gate inputs can be merged. This can result in triggers from different sources, or gates merging to create longer gates. Use the Trig/F2 module to shorten gates to allow easier stacking. Multiple wires to pitch and velocity can also be stacked, resulting in adding their values.

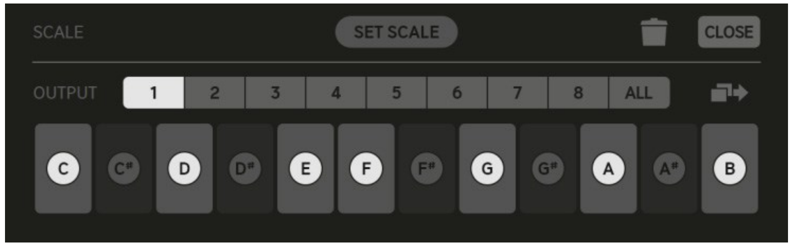
Scale





The scale module allows you to apply a unique scale to each of the 8 MIDI voices.
Use this to tailor all voices to a common scale, or get creative by filtering out unwanted notes on a per Voice basis.
For example: you can use this to so that Voice 1 sweeps an Emin arpeggio, while Voice 2 fires off a random sequence of notes picked from the whole E Minor scale, and Voice 3 plucks intervals constrained to only the root note and fifth.
Adding to the awesome power of this module is an Offset input to dynamically transpose all Voices.
This is very possibly Nest's superpower module.

Root:	Sets the root note of the scale.
Offset:	Values here transpose all voices.
Scale button:	When active, transposition from Offset will be in scale (diatonic shift). When inactive, transposition is chromatic.

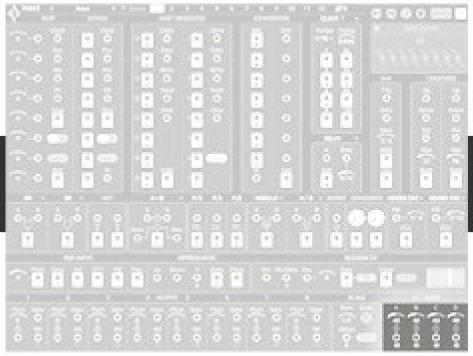


Scale Editor Window

Click the tool icon to open a floating Scale Editor Window.
Here you can set a scale for each of the 8 MIDI voices. Click on numbers 1-8 to open that voice's scale. Click on the notes to add or remove them from the scale or choose a predefined scale via the Set Scale dialog. Highlighted notes are included, and dark notes are filtered out.

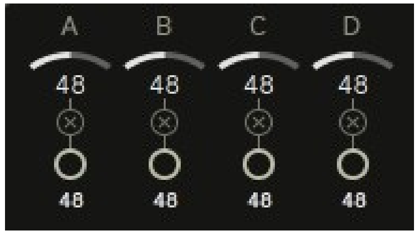
Scale buttons 1-8:	Open and set a scale for each of the 8 Voices.
ALL:	Toggle this button to apply the currently selected voice scale to all voices. (If you run into pitch issues in your project, check the state of this button to make sure it isn't accidently toggled on).
Copy:	Copy the currently selected voice's scale. Click copy, buttons 1-8 flash. Click a new voice to copy to, flashing stops, done!
Trash Can:	Will reset the currently selected voice scale to chromatic (all notes).

Tip: When sequencing external drums over MIDI, you may want to set those voices to chromatic to ensure no drum voices are accidently filtered out of scale.



CC/AUTO (Mod Out)

Like the 8 MIDI voice outputs, here there are 4 automation outputs.



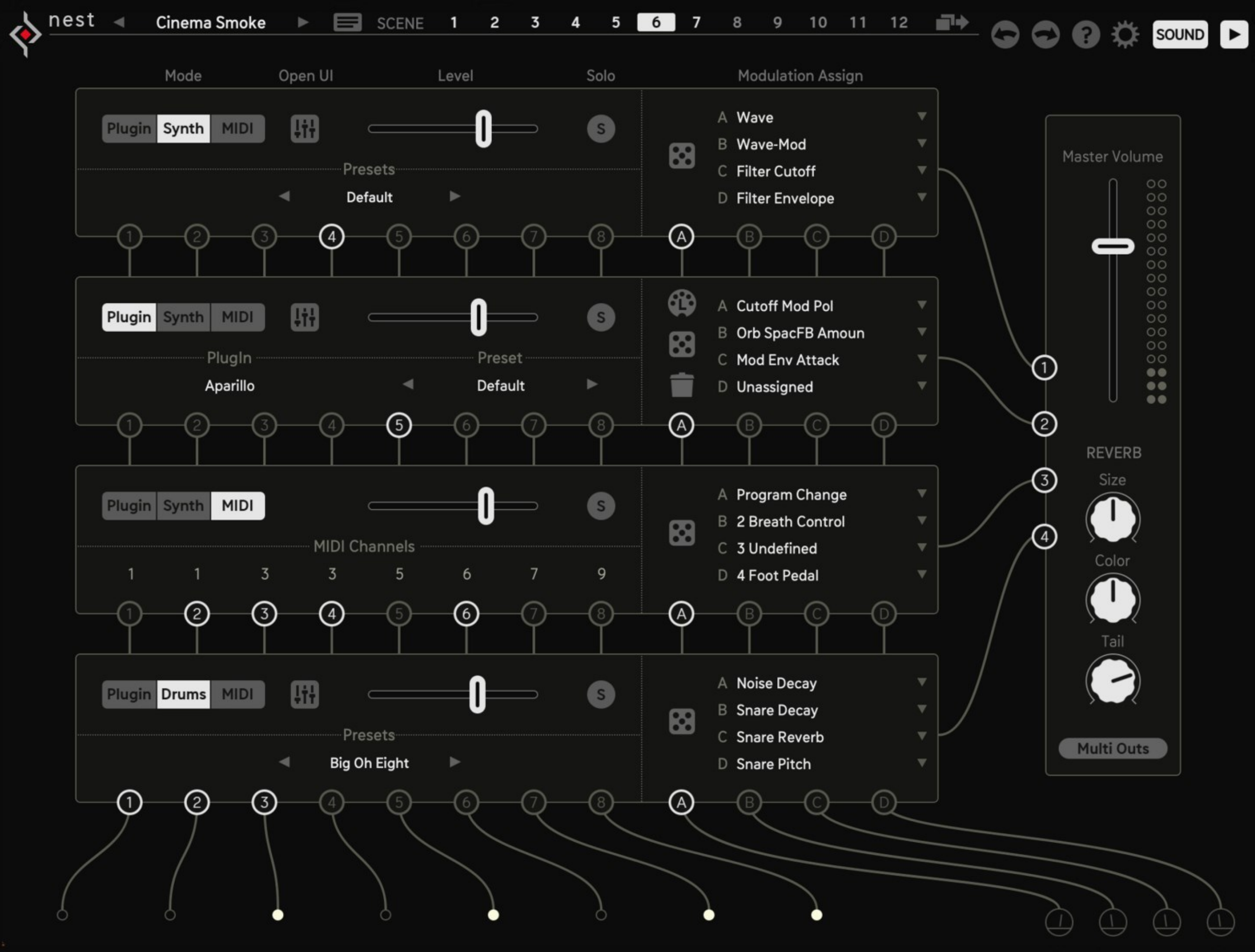
These outputs transform the values and numbers in Nest into MIDI CC, or Plug-in automation lanes.
These inputs operate in the 0-127 value range. An offset slider is provided, which sets the minimum sent value. This can be useful to get in range of certain selections or on/off parameters that operate between ranges of 63 ON and 64 OFF, (like the high-hat noise/metal switch).

VST-2 Plug-ins hosted in Nest will treat these as standard Plug-in automation. To assign a Plug-in parameter, use the MIDI Learn feature on the Sound Page.

When using external MIDI as your target, the 4 automation outputs will output MIDI CC, including Program Change and Aftertouch messages.

The Sound Page

Controls the Voice and Modulation assignment, sound generating engines, Plug-in hosting, and assigning outgoing MIDI channels

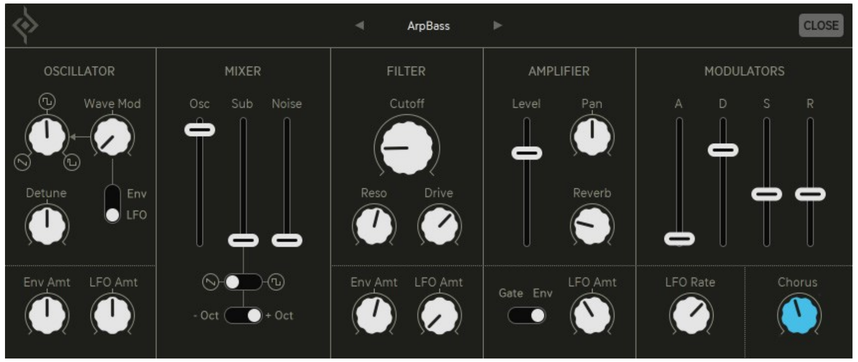


The Sound Page is where the 4 Sound Channels live. Each Sound Channel has three operating modes: Plug-in, Internal, and MIDI.
The 8 voices and 4 lanes of MIDI data from the Wire Page get converted into MIDI notes and Automation here.
These MIDI notes and automations then get assigned to the different sound channels, containing VST2 Plug-ins, sound generators or MIDI channels.

For convenience, all factory patches run on the internal sound engines. While Nest shines when you use your favorite sounds and instruments, we included these synth engines to make things a bit easier for the user at the start of a new patch, and to make sure that drums are available at any time.

Mode **Synth / Drums**

Three different synth engines are included in Nest, each engine is assigned to a different Sound Channel.

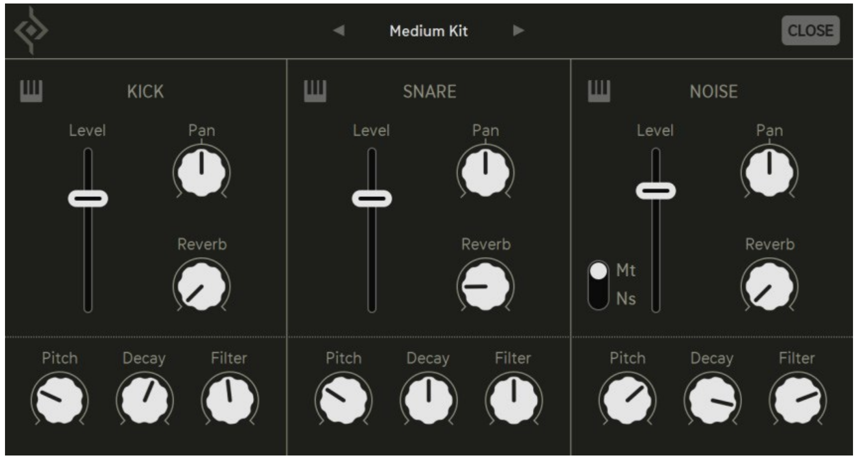


Sound Channel 1 and 2

Subtractive synthesizer, with presets. 1 oscillator with sub osc, 1 low-pass filter with drive control, 1 envelope, 1 LFO, chorus or phaser effect, and external reverb send.

Sound Channel 3

Resonator synth based on self-oscillating filter.
Percussive impulse, 3 partials with spread control, resonator with wavefolder, decay envelope, delay effect, and external reverb send.



Sound Channel 4

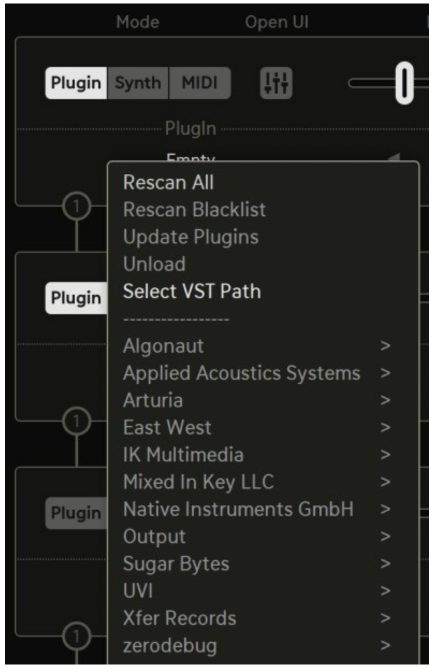
Analog drum synths with 3 channels: Kick, Snare and Noise Hi-Hat. Clicking the keyboard icon in each drum channel will enable the drum voice to listen to pitch. Otherwise, pitch is ignored.

Voice 1: Kick Drum
Voice 2: Snare
Voice 3: Noise Hi-Hat.

In Nest, the Factory Drums are assigned to MIDI Voices 1-3 (from the Wires Page). Voices 4-8 will not trigger the internal drum sounds.

Mode Plug-in

If Plug-in Mode is activated, the internal sound engine will be bypassed and a VST2 Plug-in can be used instead.



Rescan All
Rescan Blacklist
Update Plugins
Unload
Select VST Path

Algonaut >
Applied Acoustics Systems >
Arturia >
East West >
IK Multimedia >
Mixed In Key LLC >
Native Instruments GmbH >
Output >
Sugar Bytes >
UVI >
Xfer Records >
zerodebug >

The 4 parameters can be sent to the Plug-in as automation data.

Rescan All

After installing Nest, this should be your first step for integrating your VST2 Plug-ins.
Nest's scanner will focus on the VST2 installation path, unless you chose Select VST Path instead.

Rescan Blacklist

Will rescan previously excluded/crashed/unloadable Plug-ins.

Update Plug-ins

Initiates a new scan, ignoring the Plug-ins already registered with Nest – this option will save you time, when you want to include a recently installed Plug-in.

Unload

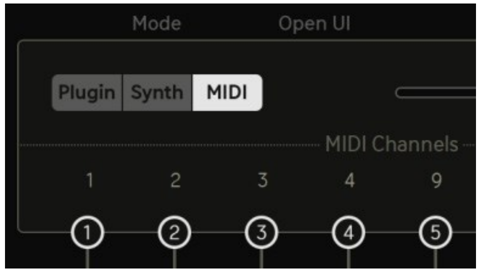
This option will remove the currently loaded Plug-in from Nest.

Select VST Path

Here, you can manually select a path for Nest's VST2 scanner.

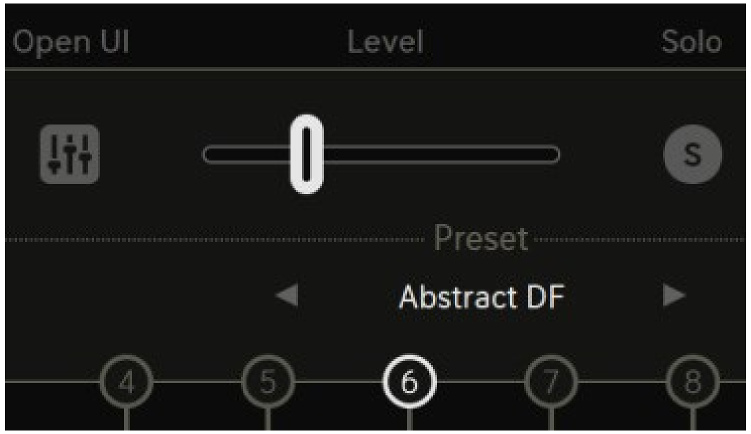
Attention! On a MacBook with Apple M1 Silicon CPU make sure to launch the related version, native ARM64 or Intel. Nest will not be able to run Intel-based Plug-ins under Rosetta 2 in a DAW that is running natively on Apple Silicon (and vice versa)! So, to use your ARM64 compatible Plug-ins you should launch Nest within a native Apple Silicon version of your DAW. To use your Intel-based Plug-ins you should launch Nest within your DAW (or the standalone version of Nest) via Rosetta. Therefore, just right-click your DAW (or the Nest standalone) icon in the applications folder and select "Open using Rosetta". This will force it to launch in Intel compatible mode.

Mode MIDI



MIDI data can be passed on to any individual MIDI channel per voice.
The 4 Sound Channels can create overlapping MIDI data (notes converging on the same MIDI channel), because the channel assignment is generic.

User Interface (Sound Channels)



Open UI
Open the graphical user interface of the selected sound generator / Plug-in.

Level
Volume of the selected sound generator / Plug-in.

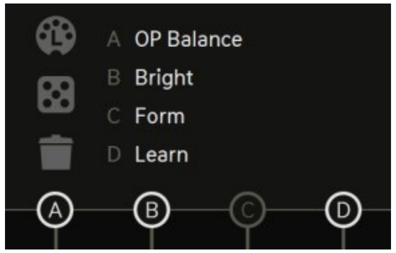
Solo
Solo a sound channel. Multiple sound channels can be soloed at once.

Presets
Presets for the loaded VST2 Plug-in, or the internal sound generators can be selected here.

Voice Assign 1-8

MIDI note data is sent from the Wire Page from Voice Outputs 1-8.
These MIDI voices can be directed to any and all of the 4 Sound Channels with the buttons 1-8.
Click on the 1-8 buttons to activate or deactivate that voice for that Sound Channel.
Voices directed at Sound Channels in MIDI mode will have an additional MIDI Channel number above the voice.
Drag up or down on that number to manually change the MIDI channel.
Data being sent over the same channel will be merged, and may lead to funny results in the case of overlapping notes.

Modulation Assign



Modulation is sent from the Wire Page to channels A, B, C and D.

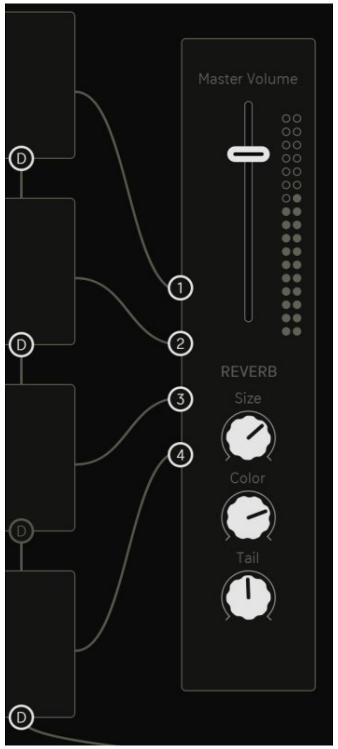
This modulation data can be directed to any and all of the 4 Sound Channels with the A, B, C, D buttons.
Modulation directed at external MIDI is converted into a user definable CC, Program Change or Aftertouch message.
Modulation directed at hosted VST2 Plug-ins can be sent to any exposed Plug-in parameter, easily assignable by MIDI Learn.

ABCD:
Activate these buttons to assign that channel of modulation to the desired Sound Channel. Each button is then represented by a menu, which offers a list of either MIDI CC numbers and names, VST2 automation parameters from a loaded Plug-in, or parameters of the internal synthesizer, depending on the selected work mode.

Learn All Button (Plug-in mode only)
The MIDI socket symbol will activate learn mode. This makes it possible to sequentially assign the 4 modulation targets to your VST2 Plug-in, just by moving a control. Click the Learn button, open the Plug-in UI, and move the desired control. It will immediately be assigned to the A target. Move another Plug-in control to assign it to the B target, and so on. To learn a specific target, right-click on that modulation target and select 'learn'.
To unlearn a target, right-click and select 'unlearn'.

Random Assign Button
Click the dice button to randomly assign 4 parameters to channels ABCD.

Unassign All Button
Click the trash icon to unassign all modulation assignments.



Output Section

1,2,3,4
These 4 buttons will activate or deactivate that Sound Channel from the output. They act as audio mutes.
In MIDI mode, these buttons will mute all MIDI output from that Sound Channel.

Master Volume
Here you can set the final output volume. If Multi-out is active (DAW only), this volume affects all Sound Channels.

Reverb
Here we have a reverb, which is on a send channel. Only the internal sound generators are sent to the reverb. Therefore, in each sound generator there is a Reverb send control. VST2 Plug-ins and MIDI mode are unaffected by reverb settings.
The Reverb offers three parameters:
Size: Controls the size of the room
Color: Damping factor. Affects the brightness and high frequency content of the room.
Tail: Length of the reverb tail / decay.

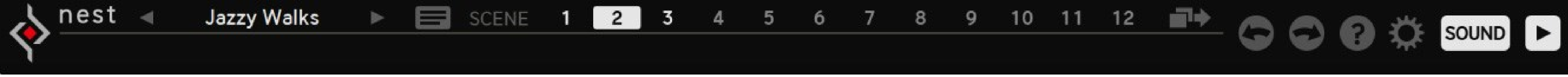
Multi Out (DAW only)
In compatible DAWs, each of the four Sound Channels and the Reverb Send can be output on their own stereo audio channel.
These Sound Channels can either be internal sound generators (subtractive, resonator, and drum synths), or a VST2 Plug-in. Nest has the ability to host Plug-ins, even in your DAW. Clicking the *Multi Outs* button will activate or deactivate this feature. By default, this is turned ON when you load an instance of Nest in your DAW. If you are not hearing sound, or don't plan on using this feature, deactivate *Multi Outs.* This Multi Out feature is useful for mixing and recording purposes, and for applying your own audio and effect chains to the sounds in Nest. Refer to your DAW's documentation for instruction on setting up multi out channels.

The multi out channels are as follows:

Nest 1-2:	Sound Channel 1
Nest 3-4:	Sound Channel 2
Nest 5-6:	Sound Channel 3
Nest 7-8:	Sound Channel 4
Nest 9-10:	Factory Reverb Send, Wet Signal

Header & Settings Screen

Manage your presets, create notes, design up to 12 wiring scenes, and set your overall project and interface preferences.



Header

Preset Name
Displays the currently selected preset. Using the left/right arrows will open the next or previous preset in the current folder.
Clicking the preset name will open a menu containing all factory and user folders. Load and Save presets from here.

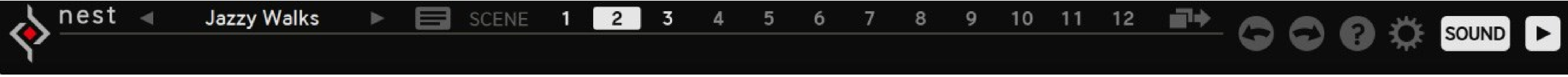
Notes
This is a note pad for you to document the construction of your project, or any important performance notes. Use this to organize your thoughts or leave reminders for yourself when returning to your presets. You might find instructions on how to use a patch here too.

Wiring Scenes 1-12
Here is what may be considered Nest's most powerful feature - the twelve Wiring Scenes. Each scene may contain a completely unique patch construction, which includes every control on the Wire Page, and all of the Voice and Modulation destinations on the Sound Page.
Scene switching is beat-synced to 1/16th notes, and can be switched manually, or by MIDI notes.
The first octave of the MIDI range (pitches 0-11) are reserved for the scenes.
This of course means that the scenes can be automated from your DAW, or hardware sequencers and controllers.

Copy Scenes
Copying scenes is simple. Merely click the Copy button, and all scenes will start blinking. Then click once on the target scene, blinking stops, success!
Among the almost infinite possibilities the scenes offer, this is also a great method to build up sequences: Start with drums, copy. Add chords, copy. Add a melody, copy, etc.

Undo/Redo
For when you've made a mistake, or accidently removed a wire. These arrows will undo or redo the last operation. Many stages of undo/redo are possible.

? (Help)
When help is active, hovering over a module header will display an info screen detailing instructions and operational info about that module.
Useful in a pinch if you've forgotten what a specific input does, or need an idea about how to use a module.



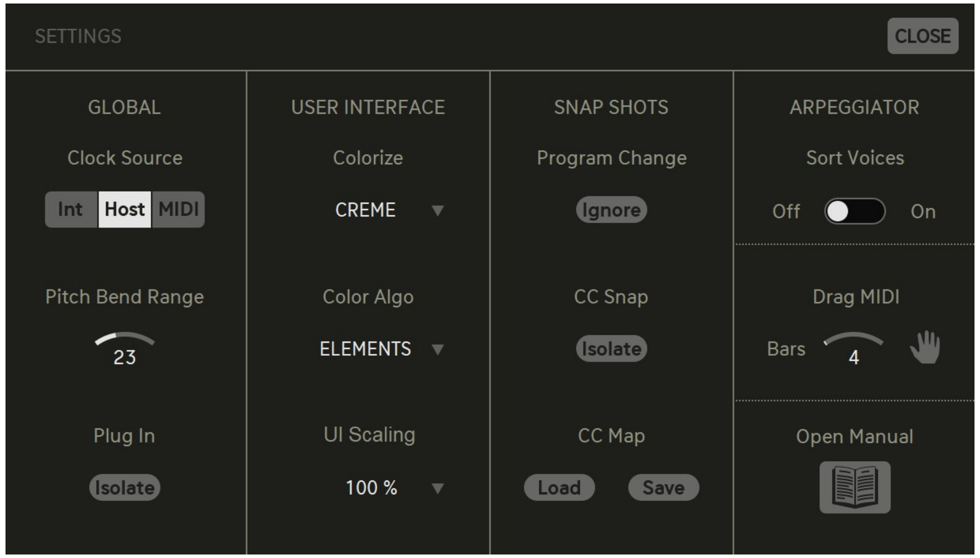
Gear Icon (Settings)
Opens up the Settings Screen where you can find global settings and functions.

Sound
Toggles between the Sound Page and the Wire Page.

Play Button
When the play button is lit, indicates that the clock is running. Check the different clock start options here.

Settings Screen

Offers some global settings and functions.


SETTINGS CLOSE

GLOBAL: Clock Source Int [✓ Host] MIDI; Pitch Bend Range 23; Plug In Isolate

USER INTERFACE: Colorize CREME; Color Algo ELEMENTS; UI Scaling 100 %

SNAP SHOTS: Program Change Ignore; CC Snap Isolate; CC Map Load Save

ARPEGGIATOR: Sort Voices Off On; Drag MIDI Bars 4; Open Manual

Clock Source

Determines Nest's clock source, and transport controls.

Internal: Nest runs on its internal clock (bpm still set by host). Transport is activated internally, and is only retriggered by the host's transport.

Host: Nest will sync to the host's tempo, and transport controls.

MIDI: Nest's transport will start only when Nest receives a MIDI note.

Pitch Bend Range

The Range parameter defines the pitchbend range of the MIDI - pitchbend wheel. The pitchbend signal is available at the MIDI Input module.

Plug-in Isolate

You can unchain the Plug-in from being recalled. This makes it possible to change Nest presets without losing the Plug-in and its related settings.

Colorize

Use these presets to set the color intensity. They range from a desaturated gray scale (frost) to a fully saturated RGB preset (pure).

Color Algo

Select a unique wire color palette using the Color Algorithms. It selects from a curated list of RGB relationships, to set the tonal mood of your preferences.

UI Scaling

Adjust the Nest user interface to your preferred size.

Program change (Ignore)
Only presets from the folder “MIDI Programs” will be used for program changes. Incoming program-change messages will be ignored.

CC Snap Isolate
When active, MIDI Learn settings will be kept and not be changed when selecting a different preset.

CC Map
Load or save your current MIDI CC assignment. You can use this to save different controller templates that might be necessary for different presets.

Arpeggiator Sort Voices
There are two options in how the voice stack is read by the counters. The Voice Stack can be sorted or unsorted.
When Sort Voices is active, voices will be sorted by their Pitch value. If the Voice Counter counts up, Pitch will also go up.
When Sort Voices is inactive, voices will be recorded in the order in which they arrived.
If the Voice Counter counts up, this will resemble the order of notes as they were played.

Drag MIDI
Drag MIDI will render the current preset as a Type X .mid file. The MIDI data of the 8 Voice Assignments to the 4 Sound Channels, and CC Mod Channels will be rendered to the MIDI file. Choose the file length with the ‘bars’ setting. Click and drag from the Hand icon to create the file.
And drop it in your desired location. Once you’ve rendered the sequence, it can then be transferred to other MIDI sequencing software, hardware, or DAWs.

Open Manual
Will open this manual from its default install location.

Host Integration

Nest works as a VST2 / VST3 / AU / AAX Plug-in in most common DAW hosts.
Make sure that Nest is installed in the Plug-ins folder used by your DAW.
Some hosts need to perform a Plug-in rescan when a newly installed Plug-in is launched for the first time.
Please find all detailed instructions for your DAW below.

Cubase

Perform a full rescan in the Cubase Plug-in manager. Make sure that the Nest.dll/Nest.vst3 (Win) or Nest.vst/Nest.vst3 (macOS) file is contained in Cubase's assigned VST Plug-ins folder. Load Nest as Instrument into an Instrument track.

Ableton Live

When using macOS, we recommend using the VST version of Nest in Live. Please note: With Ableton Live, Plug-ins will sometimes be marked as unloadable and not rescan automatically. If this happens to be the case, force a rescan by unchecking and checking the 'Use custom VST Folder' checkbox in Preferences/File Folder/Plug-In Sources. Set up a MIDI track and insert Nest as instrument.

Bitwig

In Bitwig, select Nest from the System VST Plug-ins location. Go to the Device category, Nest will be added as an Instrument Device.

Pro Tools

When installing Nest, select AAX Plug-in format. (AAX 64 is supported by Pro Tools 11 and higher.) Use Nest as an instrument.

Studio One

Go to the Studio One menu and choose Options. Click Locations, then VST Plug-Ins. Click the Add... button and select your VST Plug-ins folder. Press OK. Create an instrument track and pick Nest up from the Instrument list.

Sonar

Make sure that NEST is installed in the VST Plug-ins folder. Insert Nest as a soft synth.

Logic

Select Nest as an AU-Instrument from the I/O dialogue of a Software Instrument track.

FL Studio

Go to Channels>Add one>More...There you should find Nest and perform a refresh. You can now open Nest in the Mixer Inserts.

Installation

Download (requires login) the latest version here. Double-click the installer and follow the instructions provided by the installation process.

The standalone version and **manual** will be installed into:

Windows:	C:\Program Files\Sugar Bytes\Nest
macOS:	/Applications/Sugar Bytes/Nest

Presets will be installed into
\Users\YOURLOGINNAME (your home folder)Documents\Sugar Bytes\Nest
Important: Do not relocate Nest presets after installation!

macOS

Default installation paths:

VST2	/Library/Audio/Plug-Ins/VST/
VST3	/Library/Audio/Plug-Ins/VST3/
AU	/Library/Audio/Plug-Ins/Components/
AAX	/Library/Application Support/Avid/Audio/Plug-Ins

Windows

Default installation paths:

VST2	C:\Program Files\VSTPlugins
VST3	C:\Program Files\Common Files\VST3
AAX	C:\Program Files\Common Files\Avid\Audio\Plug-Ins

Uninstallation

To uninstall Nest, please follow the instructions:

Windows
Uninstall Nest under Control Panel>Add/Remove Software.

macOS
Delete all the following files and folders:

/Applications/Sugar Bytes/Nest
/Library/Audio/Plug-Ins/VST/Nest.vst
/Library/Audio/Plug-Ins/VST3/Nest.vst3
/Library/Audio/Plug-Ins/Components/Nest.component
/Library/Audio/Plug-Ins/Components/NestMidiFX.component
/Library/Application Support/Avid/Audio/Plug-Ins/Nest.aaxplug-in

~/Documents/Sugar Bytes/Nest
~/Library/Preferences/com.sugar-bytes.Nest.plist
~/Library/Preferences/SugarBytes/3rdParty (Disregard this file, if you also are an owner of Sugar Bytes Obscurium!)

Authorization

The serial number is requested during installation. If the serial number is missing or incorrect, the product will not produce any sound. Check the About Screen (click the SB logo) of Nest to verify whether your serial number is VALID. Manually entering the correct serial number there or performing a quick reinstall, should fix any issues. One License covers both the macOS and Windows version and can be activated for two computers at the same time. For use on more than two computers, please buy an additional license.

Appendix

Truth Tables

These tables show the all the possible logic combinations and their outcomes.
At the beginning, it may seem necessary to reference the tables. But eventually the rules will become more familiar until the functions are memorized.

Multiplexer / Demultiplexer (B mode)

Input States				Select Step
Inhibit	A	B	C	
0	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	1	1	0	3
0	0	0	1	4
0	1	0	1	5
0	0	1	1	6
0	1	1	1	7
1	x	x	x	0

OR

Input		**Output**
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	1

XOR

Input		**Output**
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	0

NOR

Input		**Output**
A	**B**	
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

Input		**Output**
A	**B**	
0	0	1
0	1	0
1	0	0
1	1	1

AND

Input		Output
A	**B**	
0	0	0
0	1	0
1	0	0
1	1	1

NAND

Input		Output
A	**B**	
0	0	1
0	1	1
1	0	1
1	1	0

Comparators

Input		Comparison			
A*	**B***	A>B	A<B	A=B	A!=B
0	0	0	0	1	0
0	1	0	1	0	1
1	0	1	0	0	1
1	1	0	0	1	0

*comparator inputs recognize all numbers values.

Contact

Sugar Bytes GmbH | Made of passion

Greifswalder Str. 29 | 10405 Berlin, Germany
phone:+493060920395

Sugar Bytes are:
Rico Bernhardt & Robert Fehse
Vincent Miethe
Nadine Purrmann