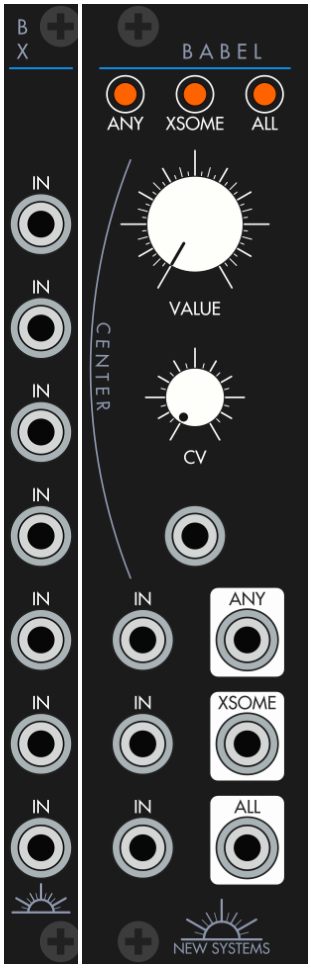
BABEL



Analog Logic and Modulation

Manual Revision 1.0.1

NEW SYSTEMS INSTRUMENTS

SPECIFICATIONS

Babel

Size	6HP
Depth	22mm
Power Consumption	+12V 44mA −12V 44mA
Output Swing	−10V–+10V peak
Input Range	−10V–+10V peak
Input Impedance	30kΩ
Output Impedance	150Ω
Output Drive	2kΩ (min), 20kΩ+ (ideal)

Babel Expander

Size	2HP
Depth	22mm
Power Consumption	None
Input Range	−10V–+10V peak
Input Impedance	30kΩ

INSTALLATION

Before installing the module, make sure the power is off. Attach the power cable to the module and to the bus. Double check the alignment of the red stripe (or the brown stripe for a multicolor cable) with the markings on the module and the bus. The red stripe should correspond with −12V, as is standard in Eurorack. Check the documentation of your bus and power solution if you are unsure. Screw the module to the rails of the case using the provided screws. (M2.5 and M3 size screws are provided.)

New Systems Instruments modules all have keyed headers and properly wired cables. But please remember to double check the other side of the cable for proper installation with the bus. Additionally, if using a different power cable, note that not every company wires modular power cables such that the red stripe will align properly with a keyed header. While our modules are reverse polarity protected as much as is practical, it is still very possible that you could damage the module, your power supply, or another module by installing the power cable improperly.

Lastly, please fully screw down the module before powering on your case. The electronics are potentially sensitive to shorts, and if the module is not properly attached to a case, there is a risk of contact with conductive or flammable matter.

INSTALLING THE EXPANDER

Each Babel expander comes with a 4 pin connector suitable for connecting it with Babel or another expander. This should be plugged such that the red wire is aligned with the "+" and the green wire with the "−" both on Babel and on the expander. The two four-pin headers on the expander are equivalent. Either one can be plugged into another expander or into Babel. Pick the one that is most conventiently located.

BASIS

Babel creates three new values from comparison of the relative magnitudes of a collection of input values, along with a center value. Specifically, given a collection **S** and a center value, C, Babel produces:

$$N = \max(\mathbf{S}); L = \min(\mathbf{S}); X = \mathrm{mid}(\mathbf{S}) = \min(\max(\mathbf{S}), C - (\min(\mathbf{S}) - C))$$

When **S** consists solely of binary values, with the higher value encoding *true*, and with C set to the midpoint between the low and high values, then these same equations can be interpreted as:

$$N = \exists\mathbf{S}; L = \forall\mathbf{S}; X = \exists_{<N}\mathbf{S} = \exists\mathbf{S} \wedge \neg\forall\mathbf{S}$$

With only two inputs, A and B, this is:

$$N = A \vee B; L = A \wedge B; X = A \oplus B$$

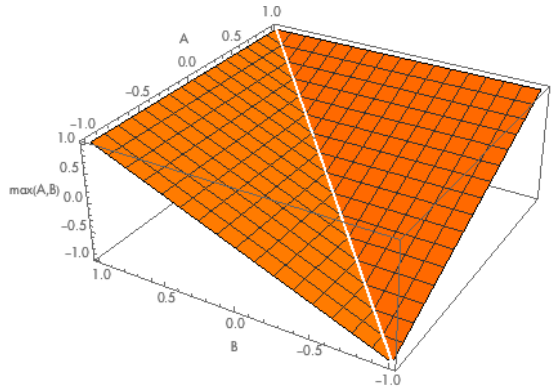
EXPLANATION

The simplest way to map between a collection of elements and a single element is to simply choose one of those elements; the simplest method of value-sensitive selection is to compare magnitudes and choose the biggest (max) or the smallest (min). Because these are fundamental functions, they provide a surprising range of complexity and versatility. But both functions are biased in one direction. This is solved with mid, which creates a version of max and min that is symmetrical around C.

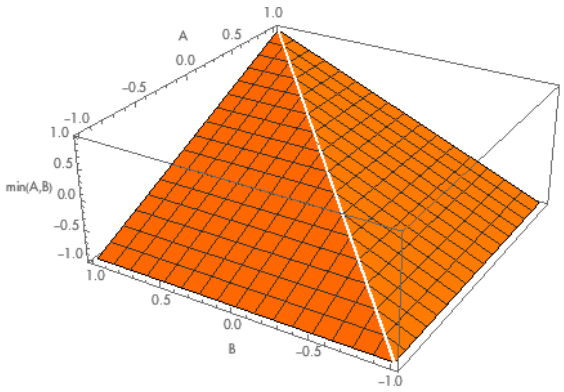
mid intermodulates two or more signals. Whereas multiplication (conventional ring modulation) produces a resulting waveform with the sum and difference frequencies of the component sine waves of the two modulators, Babel acts analogously on triangle waves, producing a mixture of triangle waves with the sum and difference frequencies of the component triangle waves that make up the original signal. Like wavefolding, the resulting waveform consists of a series of segments from the original waveshape, but sometimes flipped around C in response to the motion of another wave.

Babel conceives of logic and intermodulation as two aspects of the same thing. The max, min, and mid functions can be mapped to the fundamental logical/set theoretic functions: max ↦ ∃ (there exists some member which is true), min ↦ ∀ (all members in the collection are true), and mid ↦ $\exists_{<N}$ (some members, but not all members are true, or "exclusive some").

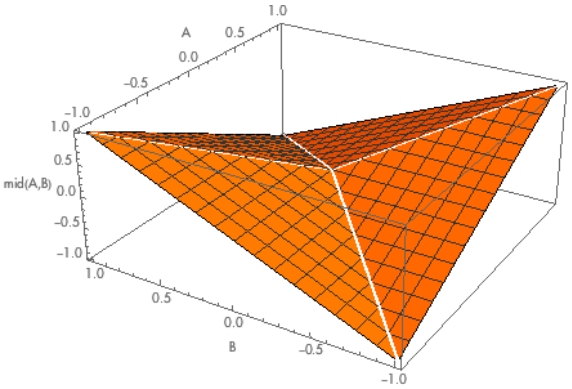
Babel's functions are named for their logical aspect, but this logic can already be direct sound, needing no interpretation, speaking in no language. Babel is a tool for building towering structures from simple discriminations.



*Max (*ANY*) function with two variables*

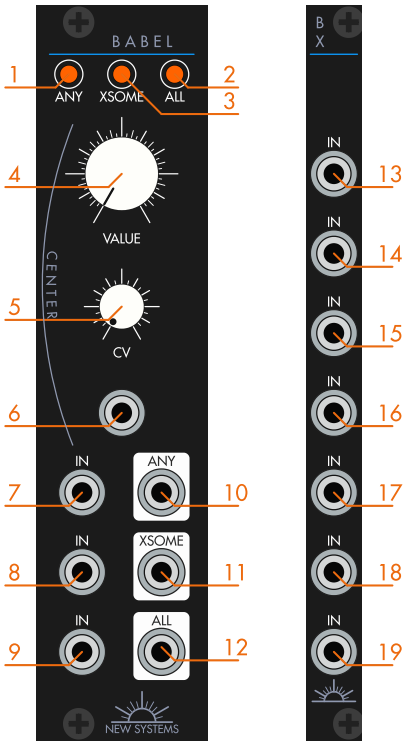


*Min (*ALL*) function with two variables*



*Mid (*XSOME*) function with two variables*

INTERFACE



1. Indicator light for ANY output. It lights when the output is greater than *C*, the center value.

2. Indicator light for ALL output. It lights when the output is greater than *C*, the center value.

3. Indicator light for XSOME output. It lights when the output is greater than *C*, the center value.

4. CENTER control. Controls the value of *C*, the center around which symmetry is preserved. All the way right is 2.5V, left is −2.5V, and center is 0V. *C* is the sum of the value of this knob and the CV value.

5. Attenuator for CENTER CV control. Ranges from 0 to 0.5, such that with the knob all the way right, a +5V input will raise *C* by 2.5V.

6. CV for CENTER control. *C* is the sum of the attenuated value of this input and the value of the CENTER knob.

7–9. Inputs, comprising **S**. When an input is not plugged, its value is not zero; it is entirely removed from consideration. In other words, the module considers only the magnitudes of plugged inputs.

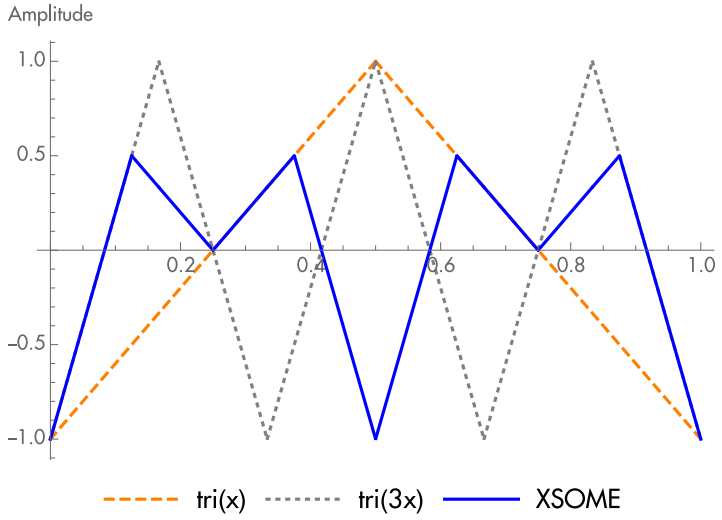
10. ANY output. Outputs *N*, the maximum value in **S**, the collection of all inputs. Considered logically, outputs *true* if any inputs are *true*, *false* if no inputs are *true*.

11. XSOME output. Outputs *X*, the balanced middle value in **S**, the collection of all inputs. Considered logically, outputs *true* if at least one but not all inputs are *true*, *false* if no inputs are *true* or if all inputs are *true*.

12. ALL output. Outputs *L*, the minimum value in **S**, the collection of all inputs. Considered logically, outputs *true* if all inputs are *true*, *false* if every input is not *true*.

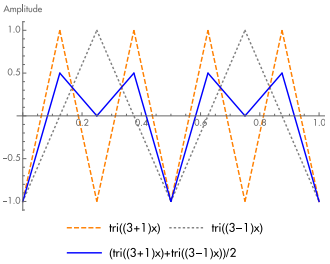
13–19. Additional inputs. When plugged, the plugged value is added to the collection **S**. These inputs are functionally indistinguishable from the three inputs on Babel.

USING BABEL FOR MODULATION



Two modulators and the resulting XSOME *waveform*

When Babel is used for modulation, it functions somewhere between a multiplier ("ring modulator") and a wavefolder. More specifically, XSOME is to triangle waves what multiplication is to sine waves. Given two triangle waves, XSOME will produce two triangle waves with frequencies that are the sum and difference of the frequencies of the original waves. Given complex waveforms, it shifts individual triangle wave components of those waveforms.

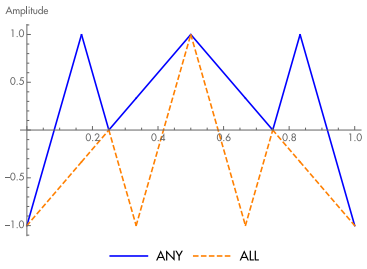


The XSOME *output as the sum and difference frequencies*

The best place to begin is by setting center to 0, plugging a harmonically rich wave such as a sawtooth, plugging a triangle to a second input, and taking the XSOME output. Tune the triangle to different frequencies and listen to the results. You will hear something like the sum and difference frequencies of a conventional ring modulator, but with a more complex, more harmonic sound.

Next turn the center dial and pay attention to what you hear. Moving the center value is sort of like moving between amplitude modulation (where the original waveforms and the new harmonics are present) and balanced modulation (where just the new harmonics are present).

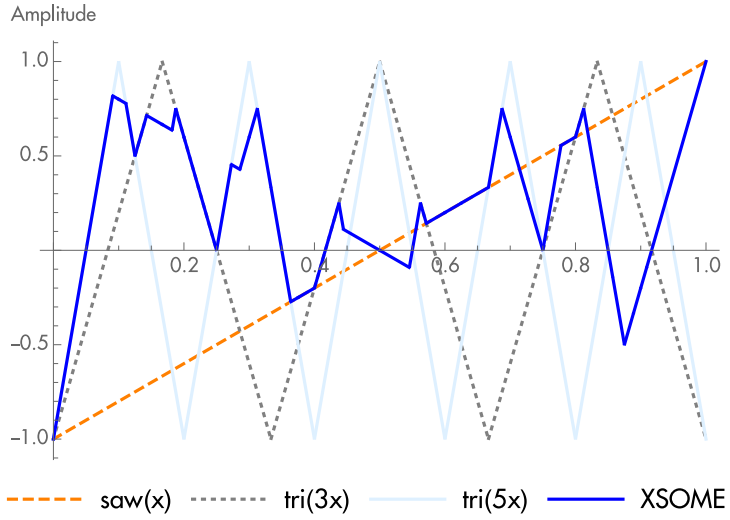
Build up the sound by plugging three waves (or more with the expander), and by plugging more complex waveforms. Try tuning these in octaves, fifths, and thirds.



*Minimum (*ALL*) and maximum (*ANY*)*

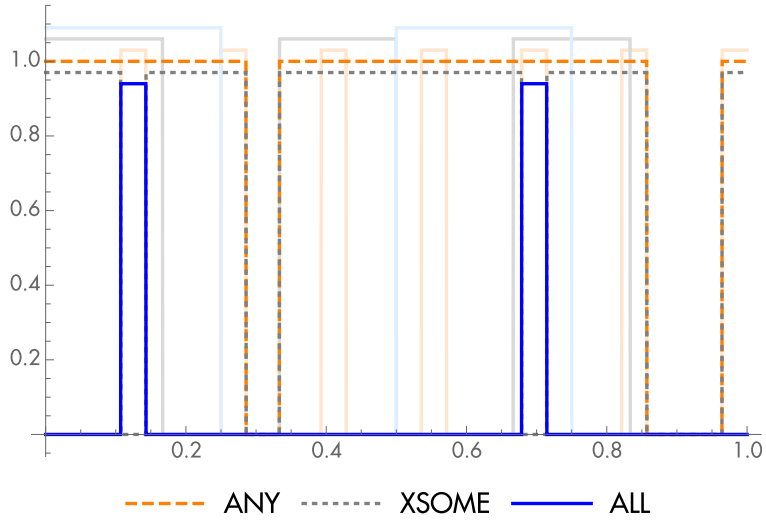
Because the minimum (ALL) and maximum (ANY) values are not symmetrical, but tend towards either low or high values, these waveforms will have a constant offset from zero. However, they are still useful modulation tools with a unique sound, particularly when mixed and/or filtered.

Lastly, fixed or slowly changing values affect the sound of the output in interesting ways. In addition to audio, there are interesting modulation possibilities with LFOS, random values, and envelopes.



Complex modulation with three sources

USING BABEL FOR LOGIC



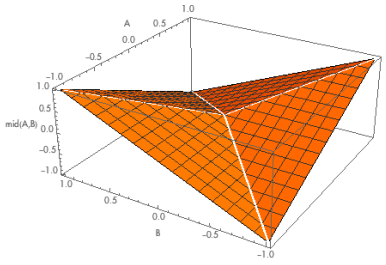
The three logical functions with three input signals

The most common logic signals are 0V *false* and 5V *true*. To work with these signals, turn the CENTER knob all the way to the right. If, however, you are using LFOs as logic signals, where *false* is −5V, turn the CENTER knob to the middle. For mixed logic, where *false* can be either −5V or 0V, again set the knob all the way right. This will output *false* values of 0V and *true* values of 5V.

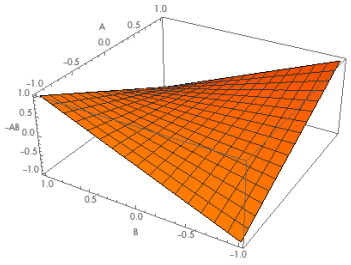
The ANY output is *true* when any of the inputs are *true*. It is the logical version of a mixer, adding multiple triggers to a single trigger output. The ALL output is true only when all of the inputs are true. It is the logical version of a VCA, where a stream of triggers is only ouput when another gate is *true*. The XSOME output is useful for creating complex sequences that are more than the sum of their parts, as, depending on the value of the other signals, any change in the incoming signal can produce any change in the ouput signal. When all other signals are true, a *true* value will cause the XSOME output to become *false* and a *false* value will cause the XSOME output to become *true*; whereas when all other signals are false, a *true* or *false* value is simply passed through to the XSOME output.

XSOME AND MULTIPLICATION

For *C* of 0 and two inputs, the mid function of the XSOME output closely approximates a multiplication. It is equivalent to a negative multiplication function, but with curves transformed into abrupt edges.

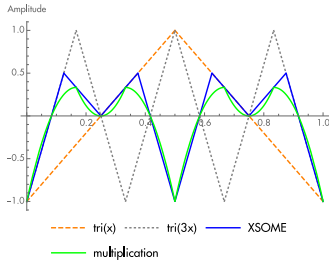


The mid *or* XSOME *function*

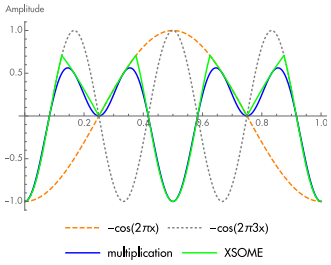


The (negative) multiplication function

As such, compared with multiplication, XSOME *generates harmonics* by generating sharper peaks, as can be seen when two sine waves are put through XSOME instead of being multiplied. However, an equally valid interpretation is that XSOME *preserves harmonics* which multiplication dampens, as can be seen when two triangle waves are multiplied instead of being put through XSOME.



XSOME *and multiplication of triangle waves*



XSOME *and multiplication of sine waves*

As is mentioned above, multiplication of two sine waves produces two new sine waves with the sum and difference frequencies of the two waves, e.g. a sine at 440Hz

multiplied with one at 300Hz will produce one sine at 740Hz and one at 140Hz. With XSOME, the exact same thing happens, but with triangles instead of sines. So applying XSOME to a triangle at 440Hz and one at 300Hz will produce one triangle at 740Hz and one at 140Hz, nothing more or less.

You may have heard that all waveforms "are made up of" sine waves of different frequencies and phases, and that these sine wave frequencies form the spectrum of the sound. This is not quite true. What we should say instead is that all waveforms *can be analyzed as* a series of sine waves of different frequencies and phases. It turns out that all waveforms can *also* be analyzed as a series of triangle waves of different frequencies and phases. The waveform itself exists in the time domain. The analysis into sine waves would be the "sine frequency domain," the analysis into triangle waves the "triangle frequency domain."

Now we can precisely characterize the relationship between XSOME and multiplication for two inputs: the effect of a time domain multiplication on the sine frequency domain is exactly equivalent to the effect of a time domain XSOME on the triangle frequency domain.