PhaMod
phase modulation synthesizer

Steven Brawer
11/22/2016

introduction

PhaMod is a monophonic, phase-modulation (or FM) plugin synthesizer, with additional amplitude modulation functionality, written in Native Instruments Reaktor 6.

My goal in creating this synthesizer is to generate persistent synthetic sounds for lyrical music. (Percussion sounds are also possible using envelopes.) My immediate goal is to use it, along with other software synthesizers and samplers, is to compose music for an opera. (See below, *issues addressed by this synth.*)

The synth is based on four operators, which can modulate each other, each with four filters in series and an AHDSR envelope. The synth includes LFO, stochastic and envelope modulations of operator phases, filter frequencies and strengths, and amplitudes. Each LFO has six oscillators. There are eight global filters in series, global AHDSR and reverb. The idea is to filter up to the level of individual partials, for very fine control of timbre (see next section).

issues addressed by this synth

It is not uncommon that an individual synthesized sound has two or more perceptibly different pitches which, at least to my ears, do not blend. This seems to be the case also for some sounds from sample-based synths, since electronic reproduction of an acoustic instrument such as a violin often has two clearly-perceived pitches. Frequently the problem can be somewhat ameliorated by vibrato or tremolo of the higher frequency.

For most conventional contemporary uses (EDM, rap accompaniment, pop songs) this is not a problem as frequently either the sound is acceptable, the individual sound does not last very long, a number of different sounds play at the same time, or time-dependence or effects are applied to the sound. But for my applications, it is not uncommon for a single instrument with a somewhat pureish (non-nasty) tone to play solo for several measures in an andante passage. If the sound has two perceptible non-blending pitches, even with LFO modulation, it is not acceptable.

A common cause of the two-or-more-frequencies effect is too prominent a fundamental, and some synths even explicitly incorporate damping or muting of the fundamental. But just as frequently, the problem involves higher pitches. Also, with the fundamental gone, the same problem may be produced by the remaining partials.

PhaMod addresses this problem in two ways. First, by having a sufficient number of bandpass/bandreject filters as well as HP filters, “offending partials” may be squelched sufficiently or other partials increased to make the sound acceptable. Second, using the same filters, almost any collection of partials may be modulated.

A second issue, just as important, involves timbre - the color of the sound. In any synth which combines two or more sound sources, relative phase can be crucial. A small change in phase can significantly modify the color, even if the gross quality of the sound is not affected. (In PhaMod, you can use this to produce stuttering sounds, briefly discussed later.) In addition, modification of one or several partials can have a significant effect on timbre. PhaMod is quite flexible in this way.

If I want a conventional sound, with whooshes, screaming trebles, deep basses, nasty tones or lots of rapid changes, there are plenty of synths which can do it. NI has a great collection. PhaMod is for a different sound.

Finally, PhaMod makes no attempt is made to try to mimic sounds of real instruments. The synth does not handle samples. There is no sequencer or arpeggiator. There are no built-in effects except for reverb. Envelopes cannot be modulated. While operators can arbitrarily modulate each other, there is no self-modulation, and in general there is no feedback.

important documentation points to keep in mind

The following are particularly important:

- The differences between the following four frequencies: global **note-on frequency**, **MIDI frequency, tuned frequency,** and the **basic frequency** of an operator.
- The use of the **ops**, **ran**, **LFO** and **env** knobs of an operator, and what can be routed to them.
- How the **LFO may be routed.**

summary of functionality

The high-level structure of the synth is shown in Fig 1 below.

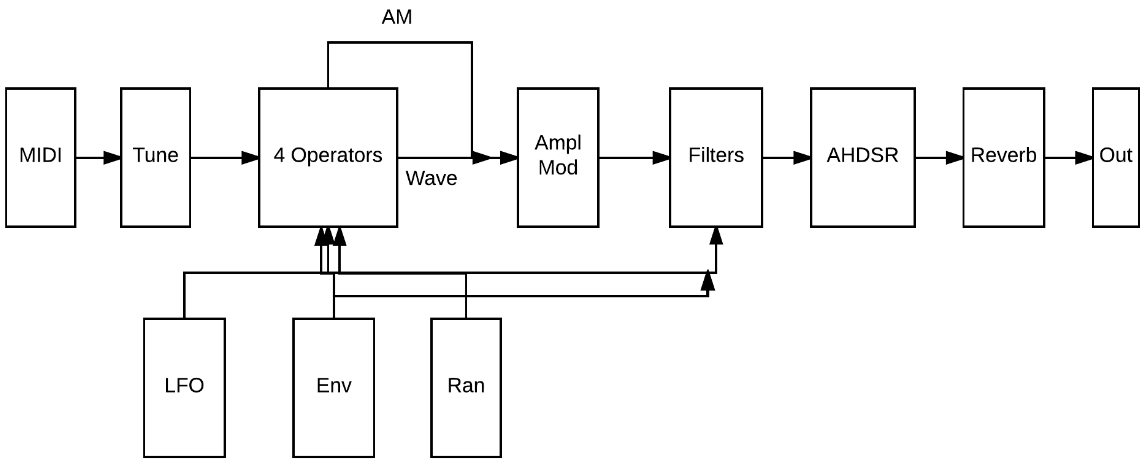


Fig 1

Operators generate phase modulated waveforms, whose amplitudes may also be modulated by each other. The LFO, Env and Ran boxes are other phase modulators. The waveform is filtered, enveloped, optionally reverbed, and output.

It is important to clarify some terminology. The MIDI note-on frequency, sent from the DAW, is called the **note-on frequency**. The MIDI box in the above flowchart receives the note-on frequency and either passes it through or modifies it. The resulting possibly-modified frequency is called the **MIDI frequency.** The MIDI frequency may or may not be the same as the note-on frequency.

The MIDI frequency can be further modified by the *tune module* in the above figure. The frequency created by the tuning may or may not modify the MIDI frequency, depending on the settings, and the result is called the **tuned frequency. S**o far we have

note-on frequency —> MIDI frequency —> tuned frequency (global) (1)

The tuned frequency is input to the operators.

The MIDI frequency is not necessarily proportional to the note-on frequency (it may be constant, independent of the note-on frequency), but the tuned frequency is always proportional to the MIDI frequency, though the proportionality may be non-linear (but always monotonic).

The operators may make further changes to create a **base frequency,** which is the actual frequency of the waveforms generated by the operators. Each operator can have its own base frequency, thus:

tuned frequency (global) —> base frequency(specific to each operator) (2)

Note: The tuned frequency also is passed to the global filters (the cutoff is based on it), though that is not apparent in the diagram above.

There are four identical operators, each generating its own waveform. The operators can be turned on or off individually. Any operator which is “on” can modulate the phase(s) of any or all of the other operators (except itself). The waveform from a given operator may or may not be sent to output (it may only modulate the phase of one or more other operators) Operator phases are also manually adjustable. In general, phases can make a significant contribution to timbre.

Each operator can independently establish its own “local” frequency either as a multiple of the tuned frequency, as a constant value, independent of the tuned frequency, or an interpolation of the two.

Each operator has two outputs, labeled **wav** and **AM**. These are identical except for destination. Each operator can output one or both. The *AM* output is, conceptually, to be used for amplitude modulation, while the *wav* output is the “normal” audio output. However, any operator can be used for both purposes. The two outputs are flexibly “mixed” in the **Ampl Mod (**amplitude modulation) unit.

The **phase** of an operator can additionally be modulated by LFOs, stochastic modules and static envelopes, indicated by the Ran, Env and LFO boxes of Fig 1.

Each operator has its own **AHDSR** envelope as well as four local **filters** in series. There is an HP filter, as well as 3 **notch** filters (a notch filter is sometimes called an **Eq filter** or a **bandpass/band reject (BP/BR) filter.)** A notch filter can function satisfactorily as an LP filter, which is why no LP filter is included explicitly.

The waveforms from those operators which are “on” are arithmetically added, possibly amplitude-modulated, and then passed through up to eight **global filters**, connected in series. These are an HP filter and 7 notch filters. The cutoff frequencies of the global and local filters can be modulated by the LFOs, the ran module and static envelopes. Further, the amplitudes of the notch filters can also be modulated by the same sources.

user interface overview

Reaktor provides two primary screens, called A and B.

<u>screen A - main screen</u>

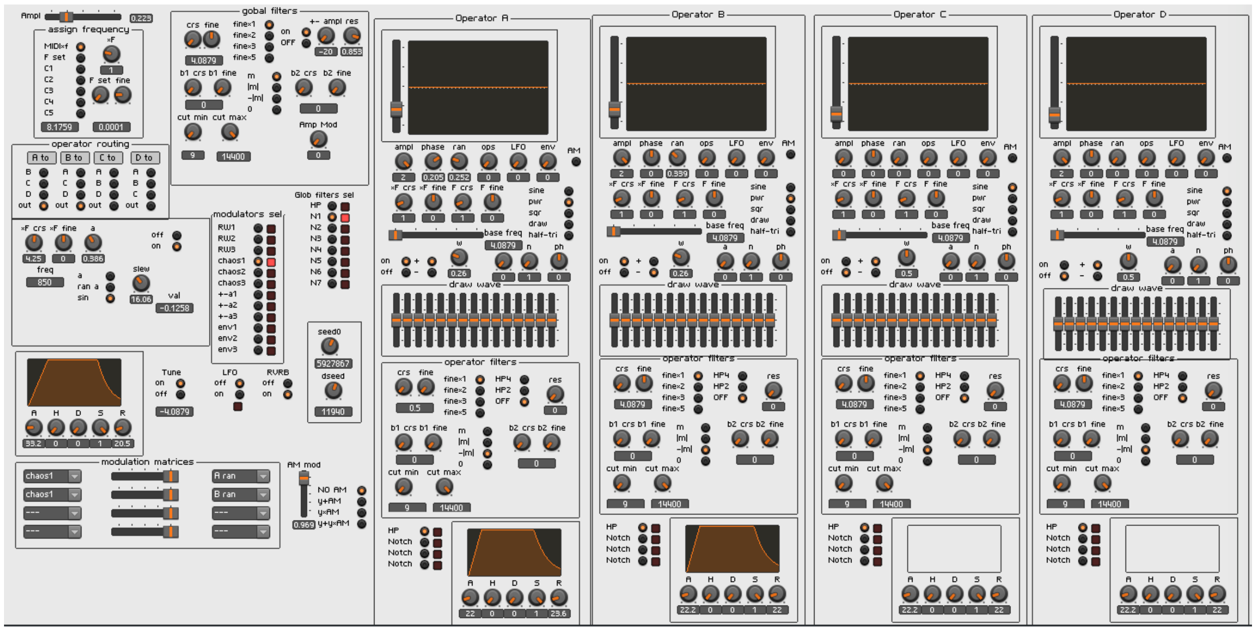


Fig 2

The four identical vertical panels on the right are the **operators**, labeled A (leftmost). B, C, D (rightmost). Each operator includes an AHDSR envelope and three filters. In this figure, operators C, D are off and so their envelopes have no image.

The fader at the top left is the overall **output volume control.**

The **input frequency control (MIDI)** is the box labeled *assign frequency* at the upper left, just under the volume control. This creates the *MIDI frequency* (see eq (1) above). Below it is the

patch panel for inter-operator modulation, Below that are the **ran** modulator modules (there are 12, all sharing the same screen space), which includes three static (non-stochastic) **envelope modulators.** Below that is the global **AHDSR** controlling the final output. At the bottom left are 4 **modulation matrices,** which dictate which modulator modulates what.

Note, as will be explained later, that *the inter-operator modulation, using a patch panel, always operates outside the modulation matrix, and the LFO modulation of operators may also bypass the modulation matrix via settings in the LFO user interface.*

At the top and to the right of MIDI and patch boxes is the **global filter** area. Each of the eight filters (one HP and 7 notch filters) occupies the same screen space. Each filter can be on or off(off = pass-through).

Below the filters and to the right of the modulators are two columns of round buttons with corresponding square panel lamps. The column on the right is to choose which filter appears in the filter box. The column on the left, next to the ran module, allows selection of which modulator appears in the ran box. Each button has an associated lamp, which is lit when the modulator or filter is active.

Just below the filter module are four small selections - a switch to turn **LFO** on and off, a switch to turn **tuning** on and off (cf eq (1)), a switch to turn **reverb** on and off, and two knobs to control the **seed** for the stochastic processes in the ran modules. The first three are for convenience, as the details are on screen B.

Finally, at the bottom right (but to the left of oscillator A), is the **amplitude modulation** section, which indicates if and how amplitude modulation should be applied.

screen B: Tuning, LFO, Reverb

The B screen is shown in Fig 3 below.

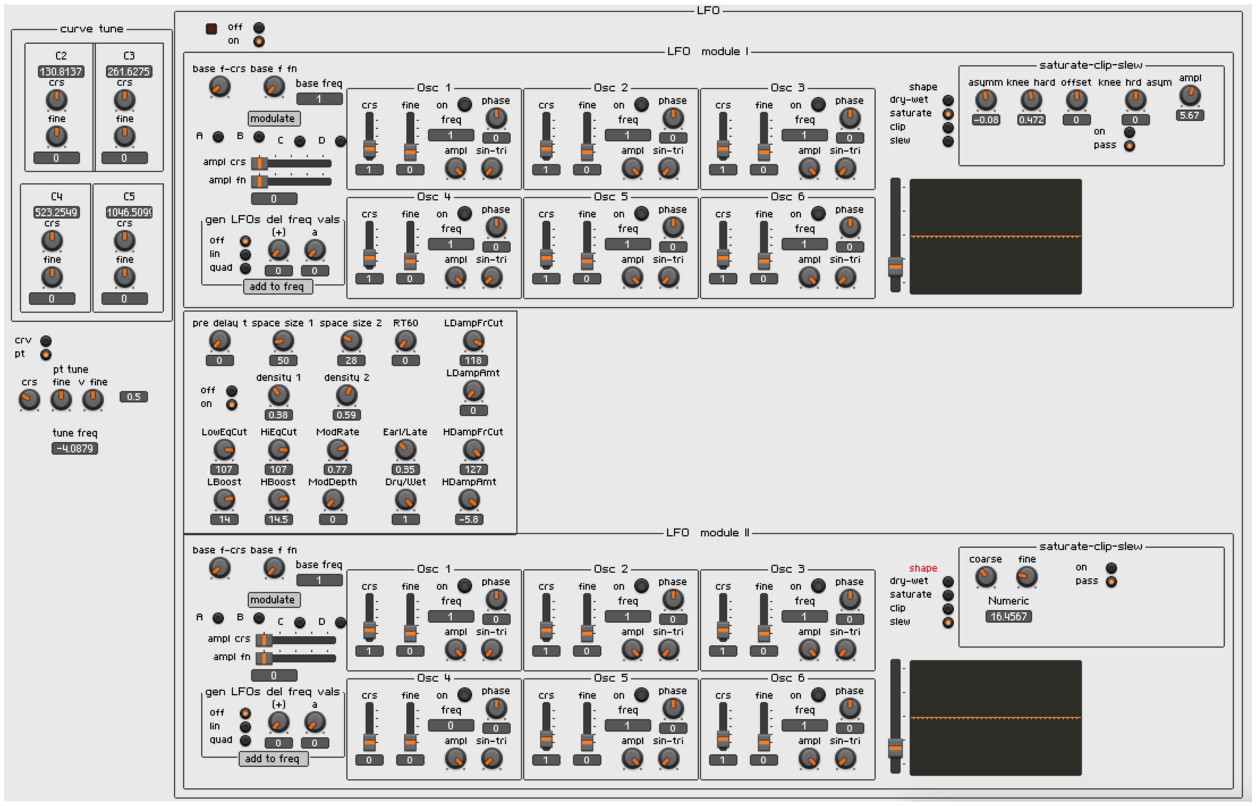


Fig 3

The narrow left section is the **tuning** section. The two wide horizontal rectangles are the two **LFOs**. The top LFO is LFO1, the bottom LFO2. Each LFO has an administrative section (on the left, which includes frequency selection and which operator(s) to modulate), six boxes representing the 6 available low-frequency oscillators, a scope showing the waveform, and a section (above the scope) for saturation, clipping and slewing of the signal.

On top of the upper LFO is a switch. The A screen and this screen both have an LFO on/off switch. Both switches must be *on* in order for the LFO to do any modulation. (If either switch is *off*, the LFO does not have any output.) There is a lamp showing when the LFO is *on*.

The box between the two LFOs is the reverb control. This is a slightly modified NI Space Master Deluxe.

MIDI module

The MIDI module is shown in Fig 4 below. The input to this module is the **note-on frequency**, and output of this module is called the **MIDI frequency (**even though ***the numerical frequency output may be different from the note-on frequency,*** as discussed previously and further below).

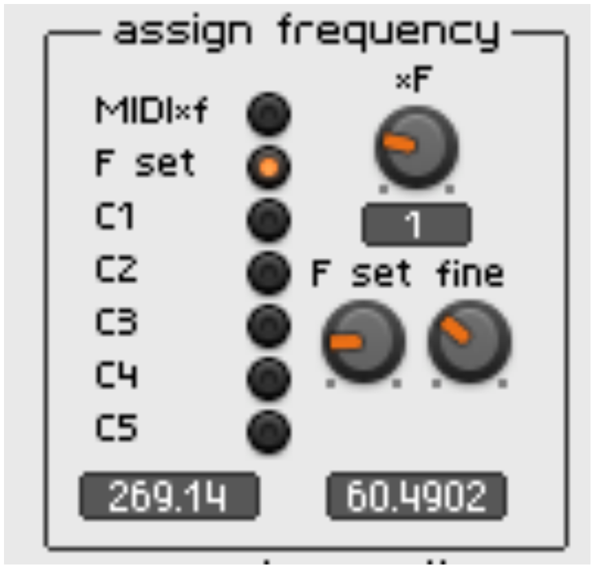


Fig 4

The two readouts on the bottom are the frequency in Hz (left) and the corresponding MIDI note number (right) output by the module. The MIDI note number is continuous (fractional), as it is computed from the log of the frequency. The frequency output by the module is called in this document the **MIDI frequency** (see eq (EQ%)).

There are seven buttons to select *MIDI frequency* and three knobs to alter that frequency.

The top button **MIDI*f** sets the *MIDI frequency* to the frequency of the note-on MIDI frequency *times* a factor set by the ***F** knob. When this knob is set to 1 (as shown in the figure), the *MIDI frequency* (the output of the module) is the frequency of the note-on frequency. When not 1, the MIDI frequency will be the value of the *F knob multiplied by the note-on frequency, and so will always be proportional to the note-on frequency.

The **F set** button (selected in the figure) allows the *MIDI frequency* to be set to any frequency, independent of the actual MIDI note-on frequency. This constant *MIDI frequency may be* adjusted by the **F set** (coarse) and **fine** buttons. The frequency is given in the readouts. This is useful for testing short-time sounds, such as percussion, across a sequence of note-on/note-off messages which may have different frequencies. The MIDI on/off gate timing and velocity are independent of these settings, but the *MIDI frequency* is constant, independent of the MIDI note-on frequency.

The remaining five buttons set the *MIDI frequency* to C1, C2, C3, C4 or C5 respectively, where C3 is middle C. The MIDI frequency is then constant independent of the note-on frequency. This is quite useful for testing audio in different frequency ranges. This is also very useful for setting tuning in situations where the synthesizer frequency is not correct, as discussed later.

patch panel for inter-operator modulation

Fig 5 below shows the patch panel.

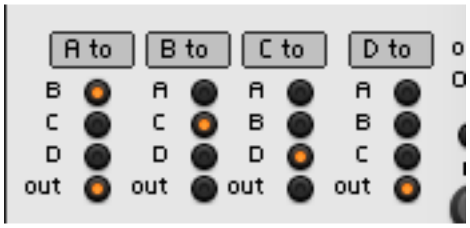
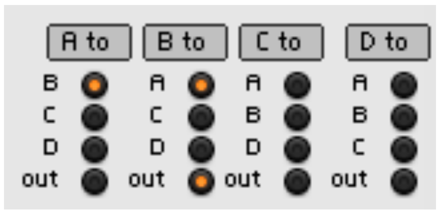


Fig 5

In the setting shown, operator A modulates the phase of operator B and also has its sound directed to output. Module B modulates the phase of operator C, operator C modulates the phase of operator D, which is directed to the output. In this configuration, the “sounds” of operators B and C are not output, so they are not heard directly.

It is possible to have feedback, so that A can modulate B, and B can modulate A, as shown below, where operator B is also output.



The strength of the modulation of an operator is determined by the combined waveforms of the other operators which are configured to modulate it, and this strength is further controlled by the **ops** knob of that operator (Fig 7 below). If the knob is 0 (all the way counterclockwise), there is no modulation by other operators no matter what the setting of the patch panel. In addition, the strength of modulation of a given operator increases with the volume of the modulating oscillators, since the modulation is based on the total output of the modulating operator.

operators

Each operator consists of three parts:

- waveform
- filters
- AHDSR

The high-level schematic for each operator is shown in Fig 6 below.

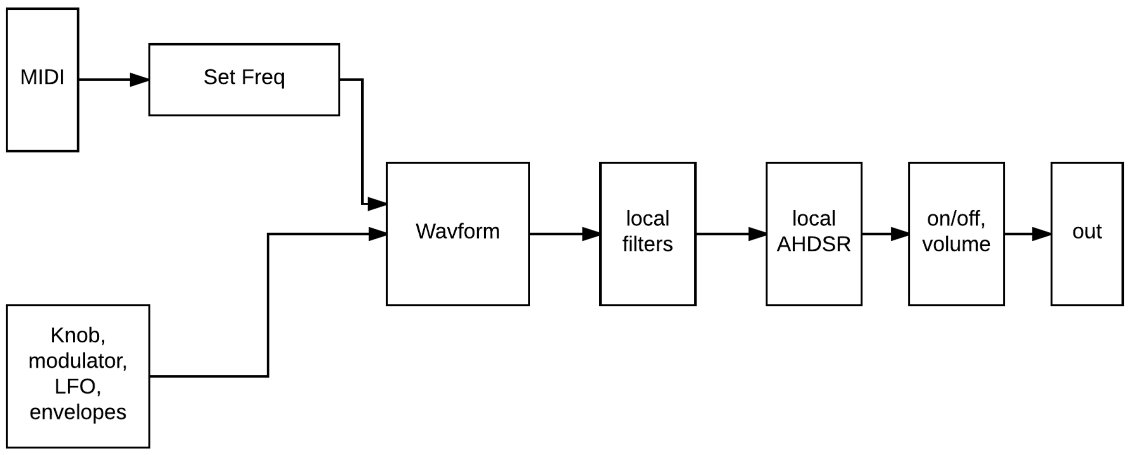


Fig 6

The operator user interface is shown in Fig 7 below.

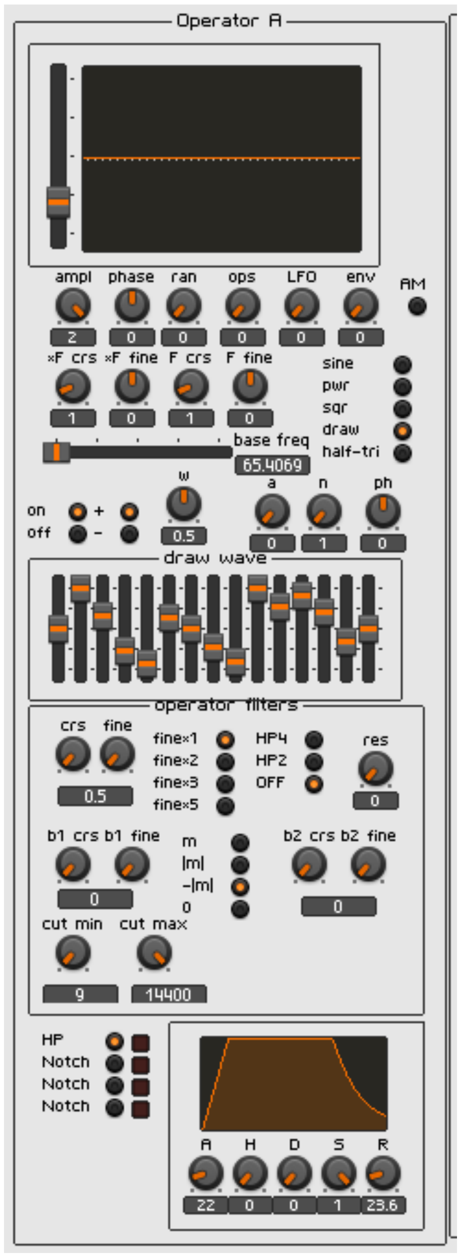


Fig 7

The scope (Fig 7) shows the *filtered* waveform.

The top, under the scope and through to the section with the 16 faders, combines the **set freq** module (Fig 6) and the **waveform generation module**. Below that is the module with the HP and 3 notch filters all sharing the same screen space. Below this, to the left, are the buttons which select which filter appears, as well as a lamp which is lit when the filter is actually filtering something. On the bottom right is the AHDSR.

waveform generation user interface

The waveform generation portion of the user interface for operator A is shown in Fig 8 below. The functionality described is common to all four operators. The technical details underlying the waveform calculation are given in the appendix.

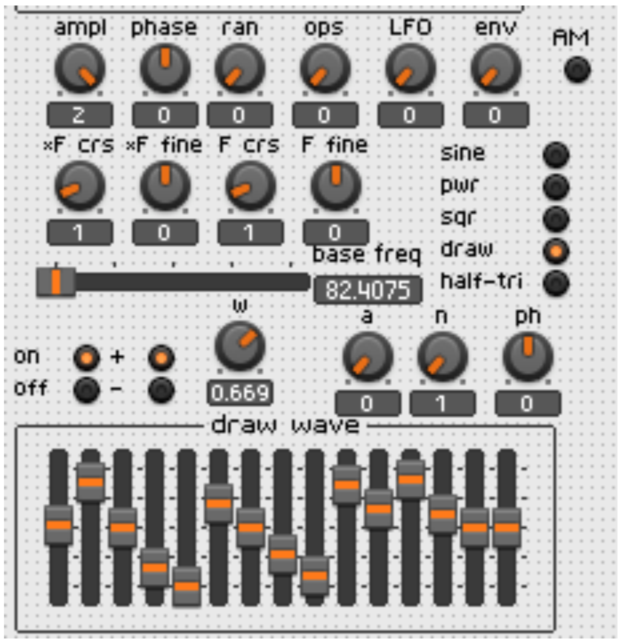
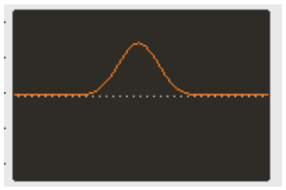

ampl 2
phase 0
ran 0
ops 0
LFO 0
env 0
×F crs 1
×F fine 0
F crs 1
F fine 0
base freq 82.4075
w 0.669
a 0
n 1
ph 0

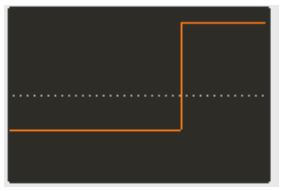
Fig 8

The **AM** button, at the top right of Fig 8, determines whether or not the operator outputs a signal for amplitude modulation (cf Fig 1). Whether or not the operator outputs an audio signal is determined by the settings in the patch panel. The **on/off switch** turns the entire operator off - when off, no sound or AM is generated. The operator can generate an AM signal even though it is not selected for output in the patch panel.

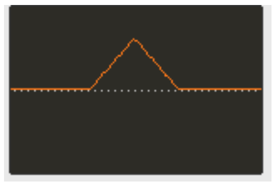
The five buttons to the upper right select the waveform. The sine wave is conventional. The **pwr** wave is sin(pi*F*t) raised to a power, and is shown below.



The **sqr** wave is conventional, shown below.

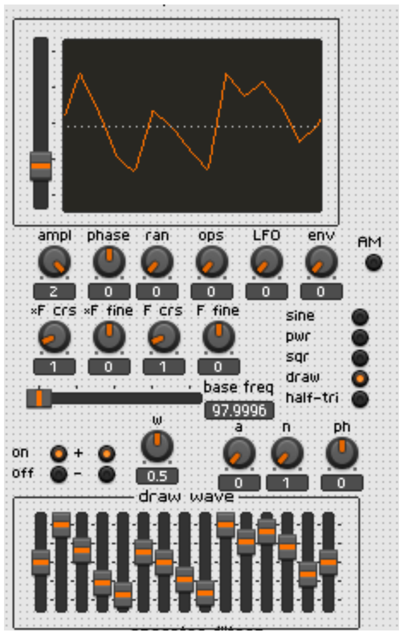


The **half-tri** is a half-triangle, shown below.



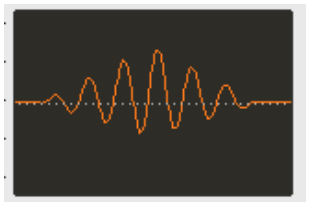
The knob marked **w** modifies the widths of these waveforms.**The larger is w, the narrower the waveform.**

The **draw** button produces the waveform corresponding to the positions of the 16 sliders, as shown below. Each slider goes positive and negative, and is 0 in the middle position.



Note how the peaks and troughs of the waveform match up with the sliders. This waveform is unaffected by the **w** knob.

The knobs marked **a, n, phs** (Fig 8) statically modulate any of the waveforms by a sine wave with **n** cycles. (The waveform is multiplied by the sine wave.) The value of **a** determines the strength of this modulation, with a value of 0 being no modulation. The **phs** is the phase of the sine wave. The figure below shows a wide (small-w) half-triangle modulated by a sine wave with 8 cycles and phase of -1.



Referring to Fig 8, several controls apply to all waveforms. The ***ampl*** knob determines the volume output by the operator. The ***phase*** knob, with values between -1 and 1, manually adjusts the phase of the waveform. Turning this knob makes the waveform in the scope move to the right or left, as appropriate. Note that a phase of -1 is equivalent to a phase of +1 (see appendix).

The +1/-1 switch inverts the waveform when set to -1. Depending on the degree of overlap of waveforms of the different operators, and on their phases, changing this setting can have a profound effect on the sound. One can use this to play some phase tricks. For example, set operators A,B with *pwr* waveform and the same w and same phase, but one is inverted, so they cancel out. Then modulate the phase of *one* of the operators (or both operators, but differently). As they go in and out of phase, they will produce a sound, and the sound will reflect the nature of the modulation.

The **ran, ops, LFO** and **env** knobs determine the strength of phase modulation by other modules. The **ops** knob controls the phase modulation strength of the arithmetic sum of the outputs of those other three operators which are configured in the patch panel to modulate this operator. The **ran** knob controls the strength of the arithmetic sum of outputs of all stochastic sources configured to modulate this operator. The **LFO** knob controls the strength of the arithmetic sum of LFOs, if configured to modulate this operator. And the **env** knob does the same for the three static modulator envelopes (in the same screen space as the ran processes).

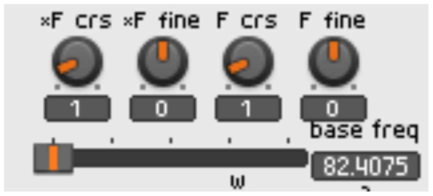
All the sources of phase modulation directed to a particular knob are added together arithmetically to produce the final modulation amount. In some cases, **the use of these knobs is a convention, not a rule.** For example, the routing matrices allow envelope modulation of operator phase to be directed to (and controlled by) the ran knob, and allows stochastic modulation of operator phase to be directed to (and controlled by) the env knob. In addition, LFO modulation can be sent to any of the env, ran and LFO knobs. (Only the ops knob is excluded from these problems.)

So these knobs are *available* to organize the control of certain sources of modulation. But the user can do what he/she wants.

Since the phase of a waveform can only vary between -1 and 1, and the modulation amounts can exceed these limits in either direction, the modulation amount of the phase is taken modulo 1 of the net value (this is explained fully in the appendix).

When a modulation knob is at 0 (fully counterclockwise), there is no modulation by the relevant sources. When the knob is at max (all the way clockwise), the modulation amount is at a maximum from the relevant sources.

The knobs and fader shown below (a detail from Fig 8) allow one to set a *local frequency* of the operator, called the **base frequency**. (See eq (2) and the discussion above it.) This is the **frequency at which the operator is actually "running".** The base frequency is read from the readout shown in the figure below. (Ignore the "w" in this figure, which labels the waveform width knob below it. See Fig 8.)



The base frequency may be proportional to the tuned frequency (and hence change when the tuned frequency changes), or be a constant value (which does not change when the tuned frequency changes), or an interpolation between the two, as is now discussed.

Let the values of these knobs (left to right) be CR*, FN*, CR,and FN (CR for coarse, FN for fine. Let **TF** be the tuned frequency(cf eq (1)). The base frequency in Hz is

$$\text{base frequency} = (1 - b)* TF*(CR^* + FN^*) + b * 200 * (CR + FN) \tag{3}$$

where the value of **b** is the value of the fader underneath these knobs, seen in the above figure. When the fader is to the left, b = 0, giving a frequency proportional to the tuned frequency, and when the fader is to the right, b = 1, and the base frequency will be a multiple of 200 Hz.Thus the slider smoothly interpolates between a frequency proportional to the tuned frequency, and a constant frequency.

The base frequency is read from the numerical output directly below the text "base frequency".

operator filters

The waveform passes through four filters, connected in series. Each filter may be turned on and off (pass-through) individually.

The global filters are the same as these local filters, so the description below will also apply to the global filters.

The group of filters of each operator are identical, and all can be modulated. Each operator has one **HP** filter, shown below in Fig 9 below.

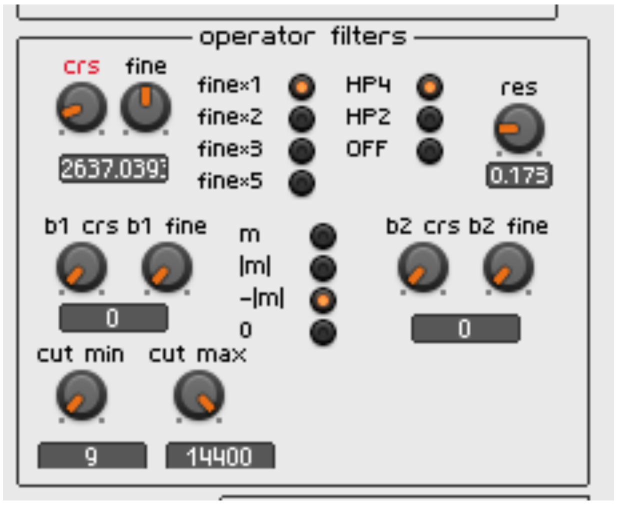


Fig 9

and 3 identical **notch** filters, one of which is shown below in Fig 10.

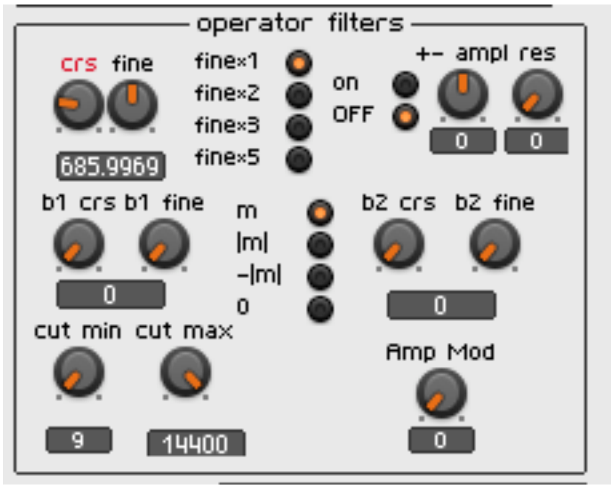
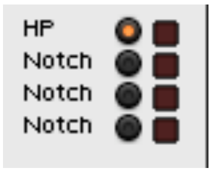


Fig 10

A notch filter is a bandpass/bandreject (BP/BR) filter.

All filters occupy the same screen space, and which filter appears is selected by the buttons under the filter, shown below.



The square next to a button is a lamp, which is lit when the filter is filtering something.

In the routing matrix, the three notch filters are labeled **notch1, notch2, notch3,** referring to the buttons top to bottom.

The controls on the HP and notch filters are mostly the same, with some slight differences. The notch filter has a simple **On/off switch**, while the HP filter has an **off/HP2/HP4** switch for the two filter modes. When a filter is off, it is a pass-through. The **res** button is the filter resonance. In addition, the notch filter has a strength-modulation capability (called **Amp Mod** in the screen) which the HP filter does not have.

In all **local** filters, the filter **cutoff frequency** is always proportional to the *base frequency*, eq (3), so that the filters always track the frequency of the waveform, which in principle can be different for each operator. (For global filters, the cutoff is proportional to the tuned frequency.)

The cutoff frequency is assigned with the **crs, fine** knobs and **fine*1,…,fine*5** buttons, upper left on the screen. The **crs** knob ranges from 1 to 30 in steps 1 (so takes on values 1,2,3,…,30). The **fine** knobs takes on continuous values between -1 and 1 if **fine*1** is selected, between -2 and 2 if **fine*2** is selected, and so forth. This sometimes makes it easier to position the cutoff frequency, especially at higher frequencies. If **cut0** is the cutoff frequency

$$\text{cut0} = (\text{crs} + \text{fine}) * \text{Fbase} \qquad (4)$$

where **Fbase** is the base frequency of the operator in Hz, and cut0 is inHz. In addition, the cutoff frequency is *restricted* to lie between 0.5 Hz and 20,000 Hz. Apart from these limits, the **cutoff frequency is always proportional to the base frequency.**

The remaining controls, except for *Amp Mod* in the notch filter, are for modulating the cutoff frequency relative to the value given in eq (4). The modulation sources include stochastic processes (ran module), envelopes (also in ran module) and the LFOs. Each of these sources generate numbers in a range [-d,d], where 0 <= d <= 1 depending on the strength of the modulators. Sometimes the numbers can be a little larger.

The **modulation amount,** called **m** in the screens, is the arithmetic sum of all sources modulating a particular filter, as configured in the routing matrix. The buttons can select **m** as is, can use the absolute value **lml** or the negative of that, **-lml.** If the 0 button is selected, there is no modulation. This is useful for turning the modulation on and off without modifying other settings.

There are 4 knobs for controling the modulation strength. The two knobs on the left, labeled **b1 crs** and **b1 fine**, make the *change* in the cutoff frequency be proportional to the base frequency, while the knobs labeled **b2 crs** and **b2 fine** make the change in cutoff frequency be independent of the base frequency. Define **B1** = b1 crs + b1 fine, **B2** = b2 crs + b2 fine, the sum of the two knobs. Let **M** be m, lml, -lml or 0, as selected by the buttons. This is the modulation amount, and is always time dependent. Then the instantaneous cutoff frequency in Hz is

$$\text{cutoff frequency} = \text{cut0} + (\text{B1} * \text{Fbase} * \text{M}) + (\text{B2} * 300 * \text{M}) \qquad (5)$$

where *cut0* is given by eq (4), the static cutoff frequency. The middle term has the cutoff modulation amount depending on the base frequency, and the term on the right sets a cutoff frequency modulation independent of the base frequency. In all cases, the modulation of cutoff frequency is time dependent due to the factor M, the modulation amount.

In addition, the cutoff frequency is always **clipped** to be between **cut min** and **cut max**, set by the knobs on the bottom of the screen. These are absolute frequencies, independent of the base frequency.

The notch filter has a button labeled **+- ampl**, which sets the **amplitude** of the peak of the filter. This knob has values in the range [-20,20]. The zero position is half-way, so to the left is band reject and to the right is band pass. The resonance makes this peak broader or narrower, as usual for BP/BR filters.

Modulation of this amplitude is controlled by the **Amp Mod** knob at the bottom right of the notch filter screen. If **Amod** is the value of this knob, then the instantaneous amplitude is given by

$$\text{notch amplitude} = \pm\ \text{ampl} + (\text{Amod} * M) \qquad (6)$$

where M is related to the modulation amount, as discussed above. The **notch amplitude** (6) is clipped to always lie between -20 and 20.

global filters

The combined signal from the 4 operators passes through the global filter section, whose user interface is in screen A, upper left, just to the right of the MIDI section. There are eight filters connected in series. Each filter has its own set of controls. All the filters occupy the same screen space, and which filter is visible depends on the buttons above the section with the filter controls.

The global filters are identical to the local operator filters, and their interface is described in the operator section of this manual. The only difference is that **the global filters calculate the cutoff frequency based on the tuned frequency, while the local filters use the base frequency of each operator.** Since different operators could have different base frequencies, the only sensible frequency for the global filters is the tuned frequency.

So to make the description of the operator-local filters apply to the global filters, replace the words “base frequency” by the words “tuned frequency”. In formulas, Fbase is replaced by TF.

tuning

The tuning section creates the tuned frequency, which is always proportional to the MIDI frequency. The user interface is mostly on screen B, with a small portion on screen A. This is the interface of screen A. This turns tuning on and off. When off, the tuned frequency equals the MIDI frequency.

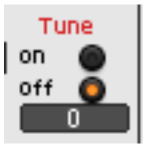


Fig 10A

The tuning section of screen B is shown in Fig 11 below.

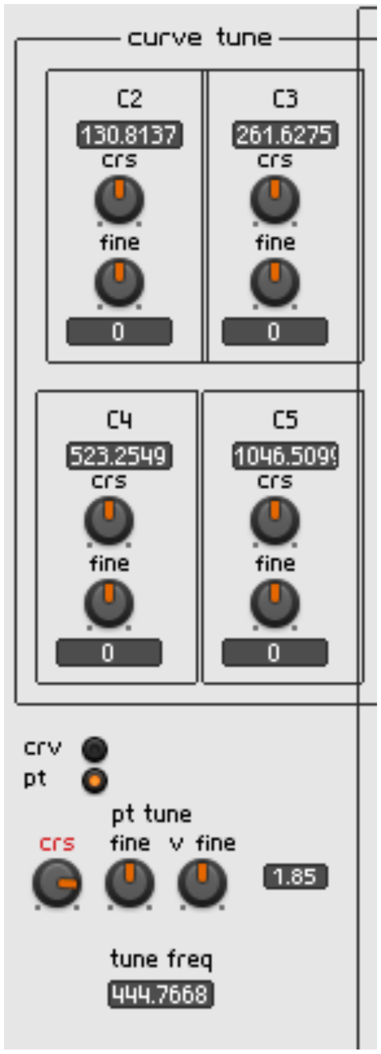


Fig 11

In general, the system calculates the tuned frequency as follows. The knobs and buttons in Fig 11 establish a small number, say T0, generally less than 10. If F0 is the MIDI frequency, then

tuned frequency = TF = T0 * F0

The quantity TF is used in eq (3), where the base frequency of an operator is calculated.

The readout in Fig 10A is the value (T0 - 1) * F0, which is the *change* (or *modification*) of MIDI frequency due to the tuning. This can be 0 in the case of no tuning (T0 = 1), negative if the tuned frequency is smaller than the MIDI frequency, and will be positive if the tuning frequency is greater than the MIDI frequency.

The **tuned freq** readout, Fig 11, gives the value *(*1 - T0) * F0, the same as the readout on the A page, Fig 10A. This is the *change* to the MIDI frequency due to the tuning. The readouts under C2, C3,… give the frequency in Hz of these notes for convenience (C3 being middle C). The other readouts give the values of the respective knobs.

There are two tuning algorithms, labeled **crv** and **pt** in Fig 11, selected by the **crv/pt switch.** The ***pt*** selection sets a single value TF for all MIDI frequencies, and is adjusted by the three knobs under the switch. These knobs allow coarse, fine and very fine tuning amounts, and the readout to the right of the fine knob is proportional to the sum of the values of the three knobs. That is,

$$T0 = (crs + fine + veryFine) \qquad (7)$$

and as usual, the tuned frequency is T0*F0, where F0 is the MIDI frequency.

The **crv** selection allows the amount of tuning to vary with MIDI frequency. This provides much greater control over the tuned frequency. If the MIDI frequency is C2 or lower, the two C2 knobs (coarse and fine values) establish the value of T0. If the MIDI frequency is between C2 and C3, the C3 knob assigns the value of T0 for a MIDI frequency of C3, and the value of T0 for MIDI frequency between C2 and C3 is interpolated between its value at C2 (set by C2 knobs) and its value at C3 (set by C3 knobs). Similarly for other frequency ranges.

The *crv* selection is useful if the frequency of the synthesizer is not correct. To use it, set the *MIDI frequency* to C2, C3, C4, C5 in turn (Fig 4) and then adjust the knobs to get the desired output frequency.

LFO

There are two LFOs, and each produces a low-frequency signal which is the sum of up to six waveforms. This signal can modulate the phases of any combination of operators as well as the filter cutoffs and notch amplitudes. The strength of the modulation for any operator selected in the LFO screen is determined by the LFO knob of the operator. It is also possible that, from the modulation matrices, the LFO can be directed to the env or ran knobs of the operator. This is not recommended, as it would likely create confusion.

There is an LFO section on both screen A and screen B, but mostly on screen B. The very simple screen A part is for **on/off** and is shown below.

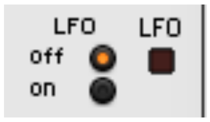


Fig 12

The **square** marked **LFO** is a lamp. It is lit if the LFO can modulate the operators and filters. If not lit, the LFO cannot modulate any operators *even if the screen-A on/off switch is on.* There is another **on/off** switch on screen B, and both switches must be **on** for the lamp to be lit and the LFO to modulate the operators and filters. Here is the screen B switch and lamp. These switches affect both LFOs.

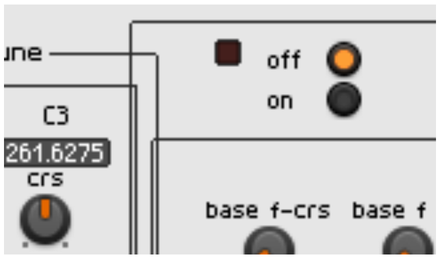


Fig 13

Screen B part has two identical LFO units (cf Fig 3), one of which (LFO1) is shown in Fig 14 below.

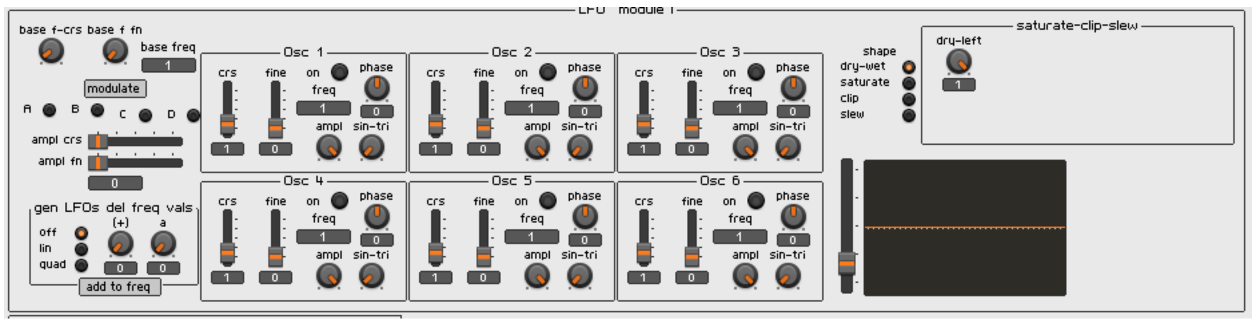


Fig 14

The leftmost section of Fig 14 is administrative, and is shown in Fig 15 below.

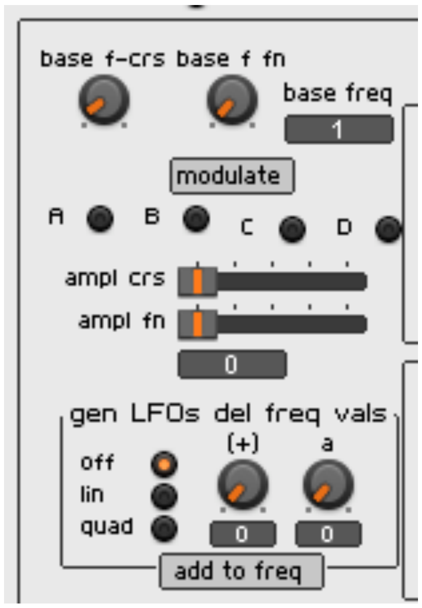


Fig 15

The top two knobs determine a **base frequency.** (This is a *base frequency* of an LFO, not of an operator.) The **base freq** readout gives this frequency in Hz. The base frequency is constant, independent of the tuned frequency. A *base frequency* of 1 Hz is shown in Fig 15. Individual LFO oscillators adjust their frequencies relative to the base frequency.

The next row of buttons selects the operators which will be modulated by the LFO. Any combination of buttons can be selected. For LFO modulation of filters, the modulation matrices must be used.

The net LFO waveform for a given LFO is the arithmetic sum of the outputs of those oscillators which are "on". The two horizontal faders affect the strength of the net LFO signal. The top fader is a *coarse* setting. The bottom fader is a *fine* setting. The readout below the *ampl fn* fader is the sum of the *coarse* and *fine* knobs. If the value is 0, there is no LFO signal.

The bottom section, unhelpfully labeled **gen LFOs del freq vals**, is a module which can assign the frequencies of all six LFO oscillators simultaneously based on several algorithms, as discussed later.

The interface for one of the LFO oscillators is given in Fig 16 below.

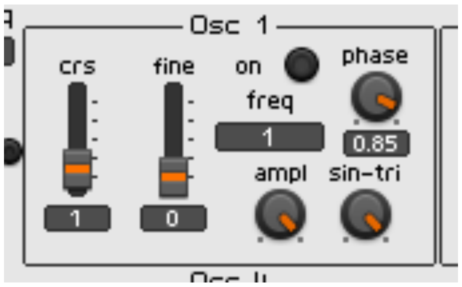


Fig 16

The button turns the oscillator on and off. The **crs** and **fine** faders, along with the **gen LFOs del freq vals** output, determine the actual frequency of the oscillator. Let **Fb** be the *base frequency* (from the administrative section). Let **Cr** and **Fn** be the settings of the *crs* and *fine* faders of the oscillator, Fig 16,and let **dF** be the output of the *gen LFOs del freq vals* section (discussed later). The quantity **dF** is an **add-on frequency**. The actual oscillator frequency in Hz is

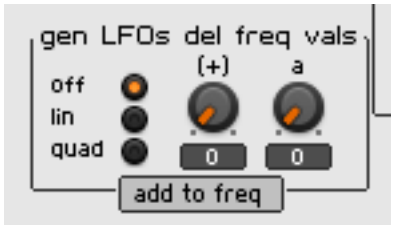
$$\text{freq} = Fb * (Cr + Fn) + dF \qquad (8)$$

So the oscillator frequency is equal to a multiple of the base frequency plus an add-on.The quantities *dF and Fb* are values in Hz, while *Cr* and *Fn* are non-dimensional values. The value eq (8) is the oscillator frequency, in Hz, and is shown by the **freq** readout of the oscillator.

Note: the dF value for oscillator 1 is always 0.

The **ampl** knob sets the amplitude of the individual oscillator, while the **sin-tri** knob interpolates the oscillator waveform linearly from sine (left) to triangular (right). The **phase** knob sets the phase of the waveform.

The *gen LFOs del freq vals* section is



which produces the dF value of eq (8). When the switch is set to *off* (the setting shown), *dF = 0.*

Let **V+** be the value of the **(+) knob.** When the switch is set to **lin** (= linear), then

$$dF = (V+) * d(n) + 0.2 * (n - 1) * a \qquad (9)$$

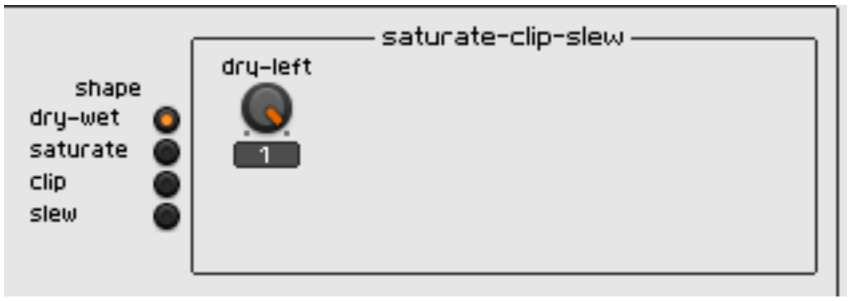
where **a** is the value of the **a knob, n = 1,2,…,6** is the oscillator number and **d(n) = 0 if n = 1 and 1 otherwise.** So this setting creates a linearly increasing set of frequencies for the oscillators, where oscillator 1 is unaffected.

When the switch is set to **quad** (= quadratic) we have

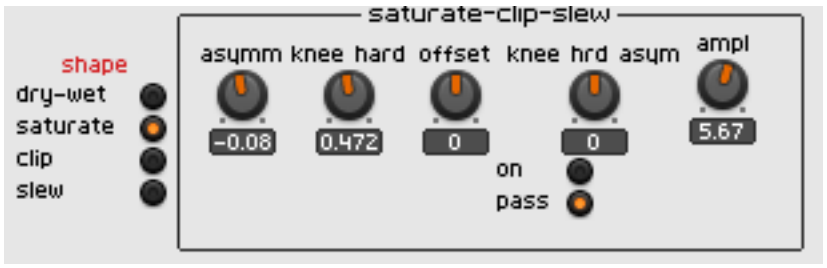
$$dF = (V+) * d(n) + 0.04 * (n - 1) * (n - 1) * a \qquad (10)$$

so the add-on frequencies are a quadratically increasing function of (n - 1).

The **saturate-clip-slew** section processes the signal produced by the 6 oscillators. There are 4 sections, *in series.* The first section is **dry/wet,** where **dry** is the unprocessed signal and **wet** is the fully processed signal, and knob interpolates between these.

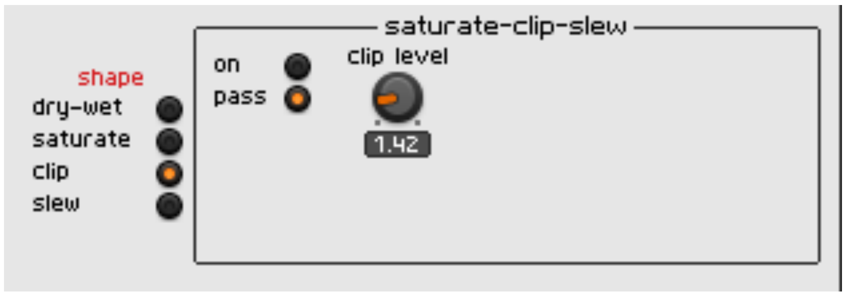


The **saturate** section is

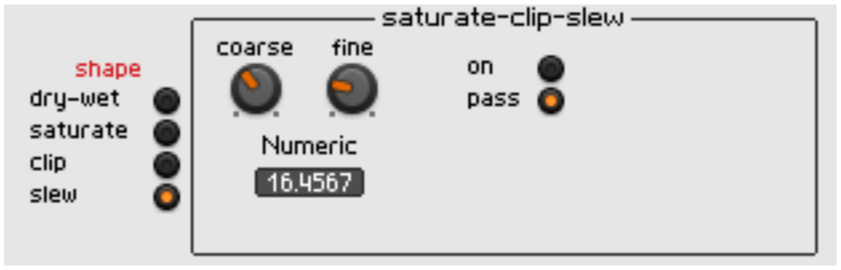


where the knobs adjust the values of the **Reaktor saturate macro**. (See that macro for documentation.) If **pass** is selected, the module is bypassed.

The **clip** section is



If the switch is on, the signal is clipped to +/- the value of the knob. Finally the **slew** section is



where the readout gives the slew value. This module controls the **Reaktor slew macro.** This macro smooths out ripples in its input. The *smaller* the coarse + fine value, the *greater* the smoothing.

The net LFO waveform is shown by the scope, where the refresh frequency is the *base frequency of theLFO (*so this image may jump around quite a bit).

ran section

The ran section has 9 stochastic modulators and 3 static envelope modulators, all sharing the same screen space. The buttons shown in Fig 17 below select which modulator appears in the box.

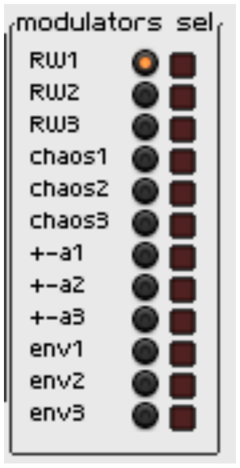


Fig 17

- **RW1,2,3** - 3 identical random walk algorithms
- **chaos1,2,3** - 3 identical chaos algorithms
- **a1,2,3** - 3 identical random value algorithms
- **env1,2,3** - 3 static envelopes

random walk RW modulator

The interface is given in Fig 18 below.

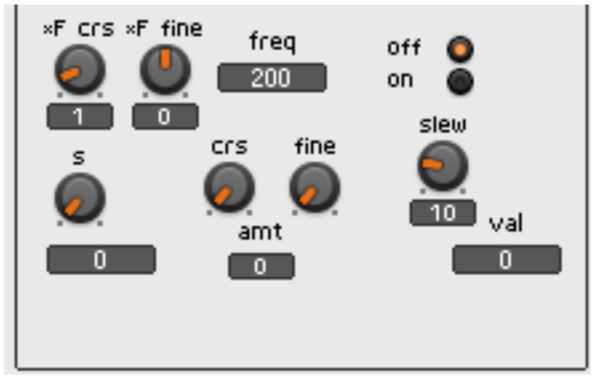


Fig 18

The random walk involves a quantity which changes stepwise at a frequency given by

$$\text{frequency} = 200 * (*\text{F\_crs} + *\text{F\_fine}) \tag{11}$$

The frequency is independent of the tuned frequency.

If **p = 1/frequency** is the period, in seconds, then there is a step (or change) every p seconds. This is discrete-time motion.

Consider the step at time step t (t = 1,2,3,….). Let **v(t)** be the value at time step t of the quantity which changes with each step, and let **R(t)** be a random number in [-1,1] generated at time step t. Then, if **s** is the **step amount** set by the **s-knob** in the above figure, and **v(t -1)** is the value of v at time step t - 1, then

$$\begin{aligned} &v(t) = v(t - 1) + s * R(t) \\ &\text{if } v(t) > 1, \text{ set } v(t) = 1 \\ &\text{if } v(t) < -1, \text{ set } v(t) = -1 \end{aligned} \tag{12}$$

Thus v(t) takes a **random walk,** step size s*R(t), at each time step t, and is bounded in value by -1 and 1. (The s knob is in range [0,1]. Note that if s = 0, nothing happens.

The output from the module is formed as follows. First, the quantity

$$A(t) = v(t) * (\text{crs} + \text{fine}) \tag{13}$$

is calculated, where **crs**, **fine** are the knobs in the center of the screen, Fig 18. This quantity A(t) is input to the Reaktor slew macro, which smooths the waveform. The smaller is the setting of the **slew knob**, the greater the smoothing. The slewed value is what is output from the module. This value goes directly to the control being modulated, as directed by the routing matrices.

The readout **val** is the slewed output of the module.

<u>chaos modulator</u>

The interface for the chaos modulator is shown in Fig 19 below. Some functionality is the same as for the RW, previous section, so that section should be read first.

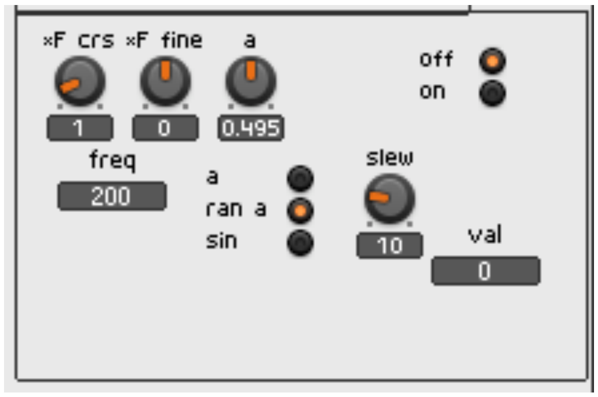


Fig 19

The chaos modulator behaves like the RW modulator, where a quantity v(t) is computed at each time step t but the calculation of v(t) is different.

The calculation uses the quantity **a**, set by the **a-knob** in Fig 19. A quantity **x(t)** is computed, based on the selected button in the middle of the screen.

- If the **a-button** is selected, then x(t) = a, independent of time.
- If the **ran a-button** is selected, then x(t) = a * R(t), where R(t) is a random number in [0,1]
- If the **sin button** is selected, then x(t) = sin(0.5 * pi * R(t) * a).

In all cases, the range of x(t) is [0,1]. Note that, for the last two cases, x(t) is time-dependent because a new random number R(t) is generated at each time step t. The random number is between 0 and 1.

Then

$$v(t) = (v(t-1) + x(t)) \text{ modulo } 1 \qquad (14)$$

where the remainder is chosen. Therefore, the range of v(t) is [0,1]. Then the quantity

$$B(t) = 2 * v(t) - 1 \qquad (15)$$

is calculated (this lies in [-1,1]), the result slewed, and the slewed result output. The **val** readout shows the skewed value.

The quantity B(t) (and hence the output) goes from -1 to 1 monotonically (but not necessarily uniformly) as the time step increases. When it becomes equal to or greater than 1, it is immediately reset to -1 (or possibly a slightly less negative value) and starts again. So it is cyclic, and increaaing a or increasing the frequency decreases the cycle time. This is different from the random walk, where the change in value can be positive or negative at each time step. Metaphorically, the chaos behavior is a randomized sawtooth.

+=a modulator

Most functionality is the same as for the RW, previous section, so that section should be read first.

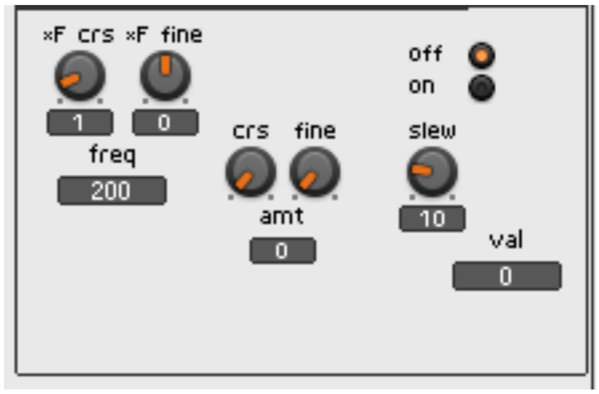


Fig 20

This is similar to the above cases, except that

v(t) = (crs + fine) * R(t),

where R(t) is a random number in [-1,1]. The value is slewed and then output.

<u>AHDSR env modulator</u>

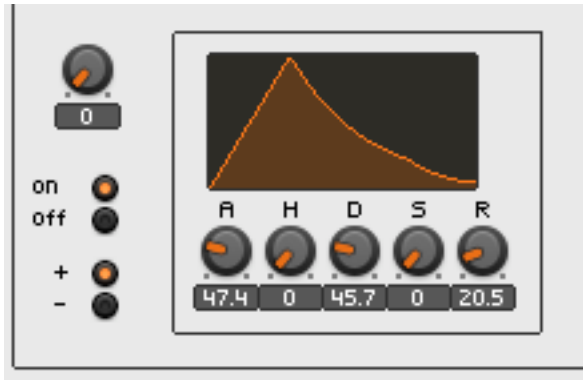


Fig 21

This modulator applies the envelope to the modulation control knobs of operators and/or filters. The envelope is gated by the note-on MIDI message. The knob sets the peak value of the envelope, while the +/- buttons invert the envelope, making it a negative modulation. The peak value of the envelope by itself is 1 and is not given by the note velocity of the MIDI gate. Therefore the actual peak value is given directly by the knob.

seed

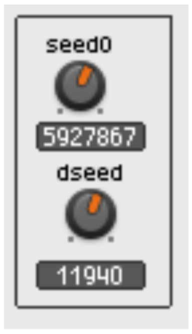


Fig 22

Each stochastic module has its own seed. The quantity **seed0** is the base seed, and the seed for a given modulator is seed0 + an integer multiple of **dseed**. There is a different integer multiple for each modulator.

amplitude modulation

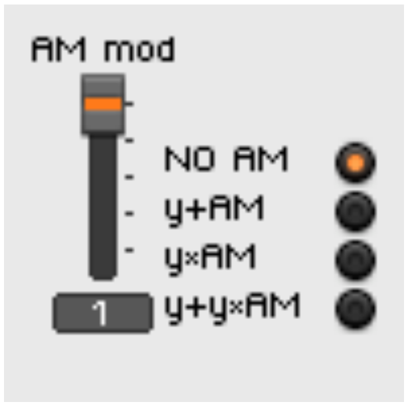


Fig 23

Let **Op(t)** be the arithmetical sum of outputs of all operators which are on and whose output is selected in the patch panel (**out button** selected). Let **Am(t)** be the arithmetical sum of **AM** outputs of all operators which are on and whose **AM button** is selected. Let **fdr** be the fader value (0 all the way down, 1 all the way up). The fader is applied directly to the AM signal. Four types of output are possible. This is the output from the **Ampl Mod** box of Fig 1. This output is passed to global filters, the global AHDSR, Reverb and global volume control.

No AM button selected (no amplitude modulation)

output =Op(t)

y + AM button selected

output = Op(t) + fdr * Am(t)

y*AM button selected

output = Op(t) * fdr * Am(t)

In this case, note that if no AM button is selected from an on operator, or if the fader i 0, there is no output.

y + y*AM button selected

output = Op(t) + Op(t) * fdr * Am(t)

modulation matrix

The modulation matrix is shown in Fig 24 below.

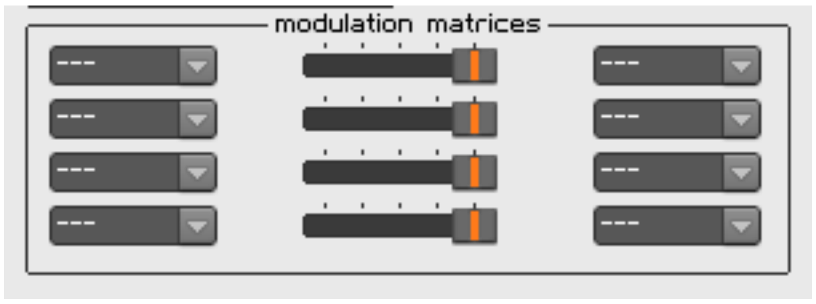


Fig 24

The left-hand side is the modulator - the source. The right side is the destination. The fader controls the strength of the modulator: left is 0, right is 1.The list of sources, which should be self-explanatory, is:

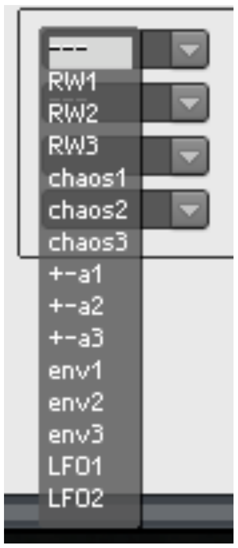


Fig 25

Here is the list of destinations.

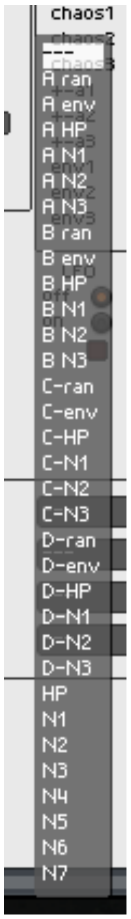


Fig 26

For example, **A ran, A env** direct the source to the **ran** or **env** knobs of operator A. Similarly, **A HP, A N1** stand for the **HP** and **notch 1** local filters of operator A. The eight destinations at the bottom are the global filters.

Note that one can send the LFO to either the env or ran knobs here, which might be confusing. One could also send a stochastic modulator to the env knob. So the use of the appropriate operator knob is a convention, and is not enforced.

appendix - waveform calculation

phasor

It might be useful to understand phasors in order to understand several features of the synthesizer. The phasor is the module which makes waveform generation possible, but it is not displayed explicitly in any flowchart (such as Fig 6).

The discussion below is for continuous time, and so is theoretical. In practice, time is discrete and audio signals vary in time at the *sample rate*, which for this synthesizer is 44,100 Hz. The implementation is not discussed here.

A phasor generates values in [0,1), which value increase linearly in time. We label the phasor output **x(t).** Let **t** be time in seconds and ***t0*** be the period in seconds (inverse of the frequency **F** in Hz of the note being played). For times such that **0 <= t < t0,** the phasor output is

$$x(t) = t/t0 \qquad \text{(A1)}$$

The expression (A1) is valid only for times such that 0 <= t < t0.

When t = t0, x(t) should be 1, but in fact x(t) is reset to 0, and so starts again from 0. Therefore, if t0 <= t < 2*t0, the value of x(t) (which is between 0 and 1) is

$$x(t) = (t - t0)/t0. \qquad \text{(A2)}$$

If 2*t0 <= t <= 3*t0,

$$x(t) = (t - 2*t0)/t0 \qquad \text{(A3)}$$

and so forth.

A simple way to express this behavior is

$$x(t) = [\ (t\ /\ t0)\ \%\ 1] \qquad \text{(A4)}$$

Here, % is the modulo binary operator, and, to avoid ambiguity, square brackets [ ] means take the remainder.

Therefore, during any particular cycle, the phasor is a linear increasing function of time. It repeatedly generates values between 0 to 1. If the frequency F is constant, the phasor is periodic with period t0 = 1/F.

There are a number of phasors in the synth. Each operator has its own phasor (since the operator base frequency may in principle be different for each oscillator). Each LFO oscillator has its own phasor, and each stochastic modulator has its own phasor. The LFO and operator phasors are all reset to 0 at MIDI note-on, so these oscillators synch. However, the stochastic phasors are not reset - they are free-running. This means that, for sounds with stochastic modulation, the timbre continually changes from note to note. This is useful as a repeating randomness can become annoying.

In all cases the frequency of a phasor is constant during a *cycle* (a *cycle* is a period of time during which x(t) goes from 0 to just before reset), but may vary from cycle to cycle. The *phase* of an operator may change continuously in time during a cycle, but not the frequency of the phasor.

To illustrate the use of the phasor in generating waveforms, using eqs (A1), (A2) and (A3) directly (and not eq (A4)), consider generating a simple sine wave (which, in the absence of modulation, produces a pure sound at a single frequency). The value of the wave is (for unit amplitude)

$$y(t) = sin(2 * pi * x(t)) \qquad (A5)$$

For values of t such that $0 <= t < t0$, we have eq (A1) or $x(t) = F * t$ (recall, $F = 1/t0$), so from eq (A5)

$$y(t) = sin(2 * pi * F * t) \qquad (A6)$$

the usual formula. Now consider times $t0 <= t < 2 * t0$. Then $x(t) = (t - t0)/t0$ (eq (A2)), and eq (EQP) becomes

$$\begin{aligned} y(t) &= sin(2 * pi * (t - t0) / t0) \\ &= sin(2 * pi * F * t - 2 * pi) \\ &= sin(2 * pi * F * t) \qquad (A7) \end{aligned}$$

the same as eq (A6). This deduction uses the following: For any value **z**

$$sin(z + 2 * pi * j) = sin(z) \qquad (A8)$$

for any positive or negative **integer j.** To get eq (A7) from eq (A8), set $j = -1$ and
$z = 2 * pi * F * t$.

We see that, given the behavior of the phasor, eq (A5) is a periodic sine wave with period t0, for all times (when frequency F is independent of time).

<u>phases</u>

Consider now the wave

$$y(t) = sin(2 * pi * (x(t)+ p)) \qquad (A9)$$

The quantity *x(t)* is the phasor output, and the quantity **p** is the **phase** of the wave, and it can be any number at all. In addition the phase *p* can have arbitrary time dependence, but for simplicity in this initial discussion, assume it is constant in time.

We can simplify things if we define a quantity **h(t)** as

$$h(t) = x(t) + p \qquad (A10)$$

so that

$$y(t) = \sin(2 * pi * h) \qquad (A11)$$

where the time dependence of *h* is now implicit.

Now suppose $1 <= h < 2$

This can happen when $p > 1$, for example. Using eq (A8), we see that we can write eq (A11)

$$y(t) = sine(2 * pi * (h - 1))$$

Suppose now that $2 <= h < 3$. Then we can write eq (A11) as

$$y(t) = \sin(2 * pi * (h - 2))$$

and so forth. Therefore, for integer *n* such that $n <= h < (n+1)$, eq (A11) becomes

$$y(t) = \sin(2 * pi * (h - n))$$

In all these cases, $0 <= h - n < 1$ for the appropriate n. In other words, for any $h > 0$

$$\sin(2 * pi * h) = \sin(2 * pi * [h \% 1]).$$

Now suppose that $h < 0$. In eq (A10), this can happen at some or all times if $p < 0$. Suppose first that $-1 <= h < 0$. Then, using eq (A8),

$$\sin(2 * pi * h) = \sin(2 * pi * (1 + h - 1)) = \sin(2 * pi * (1 + h))$$

Note that $0 <= 1 + h < 1$ in this case. This is important.

Suppose $-2 <= h < -1$. Then

$$\sin(2 * pi * h) = \sin(2 * pi * (2 + h - 2)) = \sin(2 * pi * (h + 2))$$

We have $0 <= h + 2 < 1$. In the general case, if $-n <= h < -(n-1)$

$$\sin(2 * pi * h) = \sin(2 * pi * (n + h)) \qquad (A12)$$

where $0 <= n + h < 1$ for the appropriate n.

It turns out that the modulo function has the property that, if $-n <= h < -(n-1)$, we have

$$[h \% 1] = n + h \qquad (A13)$$

(As an aside, computer languages such as C, C++, Python, Objective-C and others will give this result, which is mathematically correct. Reaktor's *modulus* macro also behaves this way. However, the Swift computer language purposefullly behaves differently.)

Based on this, and the results above, we have the very general result

$$\mathbf{\sin(2 * pi * (x(t) + p(t))) = \sin(2 * pi * [(x(t) + p(t)) \% 1]} \qquad \text{(A14)}$$

for any p(t) whatsoever.

Eq (A14) is the key equation for allowing modulation by phases. Note that, for all values of phasor output x(t), and for any p(t),

$$\mathbf{0 <= [(x(t) + p(t)) \% 1] < 1} \qquad \text{(A15)}$$

This very useful relation allows us to generalize the above discussion to any waveform at all, not just to sine waves.

In eq (A14), if p is constant, independent of time, the result is a simple sine wave, merely displaced in time by phase p. Only when p(t) is time dependent (meaning the phase of the sine is modulated by a time-dependent modulation) does eq (A14) lead to more complex waveforms.

the general waveform

Because of eq (A15), the above calculation can be repeated with any wave form whatsoever. Let **H(x)** be some function of *x,* defined for *0 <= x < 1.* The quantity *H(x)* is our general waveform, and it must be a function capable of being sensibly discretized at the sample frequency. Quite simply, if x(t) is the phasor output, the expression

$$y(t) = H(x(t))$$

is a **periodic wave at the frequency F of the phasor.**

All we really need to know is the function *H(x)* for *0 <= x < 1*. But let us extend the function, so that it is periodic. That is, for any *z*, *0 <= z < 1*, we have

$$H(z + n) = H(z) \qquad \text{(A16)}$$

for positive or negative integer *n*. As will become clear, we do not ever have to calculate *H(z)* for *z* outside the range [0,1).

The equation (A16) is the analog to eq (A8) for the sine function. We can now add arbitrary phase p(t) such that

$$\mathbf{H(x(t) + p(t)) = H(\ [(x(t) + p(t))\ \%\ 1]\ )} \qquad \text{(A17)}$$

exactly analogous to eq (A14). Eq (A17) is what is used to generate arbitrary waveforms.

In the most general case, the phase *p(t)* may be a complicated function of t. If p(t) is differentiable everywhere (for our purposes, continuous), then the overall wave has an instantaneous frequency F + dp/dt, with dp/dt the first derivative of p. This frequency can change continuously in time.

However, for stochastic modulation, p(t) jumps discontinuously with some frequency (settable in the user interface of the modulator). In this case, the frequency of the waveform is constant during each period of the stochastic modulator but changes from period to period.

Therefore, the actual frequency of the waveform may change continuously or discontinuously with time, producing a variety of effects.