



AODYO INSTRUMENTS

# anymaV

User manual

VERSION 1.3.0  
OCTOBER 31, 2024  
[www.aodyo.com](http://www.aodyo.com)



**AODYO INSTRUMENTS**

*[www.aodyo.com](http://www.aodyo.com)*

# Contents

<b>Overview</b>	<b>5</b>
<i>Minimum requirements (6) • Installation (8) • Setting up audio and MIDI settings (10) • Demo mode, license and activation (12) • First steps (14) • Playing with a wind controller (15) • Playing polyphonically (16) • Playing with a MPE controller (16) • Video tutorials (17) • Adjusting mappings (17)</i>	
<b>The matrix</b>	<b>19</b>
<i>The matrix is the interface to your patch (19) • Morphing (20)</i>	
<b>The anatomy of a patch</b>	<b>23</b>
<i>Introduction (24) • Oscillators (25) • Effects (27) • Audio signal path (28) • Mappings (30) • Modulators (31) • Velocity Envelope and expression (33) • Connecting the synth to the matrix (34) • Patch structure (35) • The bank (36)</i>	
<b>Editing patches</b>	<b>37</b>
<b>Tuning and microtonality</b>	<b>41</b>
<i>Importing tunings (42) • Preset tunings (42) • Using MTS-ESP (46)</i>	
<b>Release notes</b>	<b>47</b>
<b>Complementary information</b>	<b>49</b>
<i>Credits and licenses (49) • Trademarks (58) • Disclaimer (58)</i>	
<b>Appendix: Module reference</b>	<b>59</b>
<i>General (60) • Oscillators (78) • Effects (149) • Modulators (201) • Mappings (266)</i>	



# Overview

**Anyma V** is a powerful hybrid physical modelling virtual instrument that allows you to create and play with new sound universes inspired from the real world, where objects collide, vibrate, and resonate.

It blends the classic ingredients of electronic music with physical modelling technology, allowing it to simulate acoustic sound sources, such as strings or reeds, as well as resonating structures, like wood, glass, or metal.

If this introduction sounds familiar, then you probably know our Anyma Phi hardware synthesizer. You can think of **Anyma V** as its virtual counterpart, as it shares the same sound engine and patch format. This means you can create patches on **Anyma V** and use them on your Anyma Phi, and vice versa. The major difference between both products is that **Anyma V** can be used as a polyphonic instrument.

**Anyma V** is compatible with any MIDI controller, such as keyboards, sequencers, or even wind controllers, including our own Sylphyo. In addition, it supports MPE controllers and DAWs, allowing you total freedom in how you control your sound.

No matter your level of expertise, synth veteran or total novice, you will find that **Anyma V** provides an easy way to play a wide variety of expressive sounds and offers very deep editing and sound design capabilities, without forcing you to dive into the details if you don't want to.

You might already be familiar with several hardware or software synthesizers, and you will see that this one shares many similarities, but is also a bit different. Of course, you can play this synth using a standard MIDI keyboard or sequencer, but contrary to most synths, this one also strives to make it easy to create very rich sounds to be controlled using expressive controllers such as the Sylphyo.

This manual will guide you through all you need to know to master the ins and outs of Anyma V. Feel free to skim the chapters you're interested in, and to come back later when you need more information about any topic. If you cannot find what you're looking for, you might want to exchange with other users, or contact us directly at [support@aodyo.com](mailto:support@aodyo.com).

## Minimum requirements

Anyma V should be able to run on any computer made in the last 10 years that can withstand a music production workflow. It runs on Windows, macOS and Linux, and is available both in standalone and plugin formats.

### Windows

We tested Anyma V mainly on Windows 10 and 11 computers.

#### **Processor**

Intel/AMD (32-bit or 64-bit) or ARM 64-bit processor

#### **RAM**

1 GB of RAM

#### **Disk space**

100 MB of free disk space

#### **Operating system**

Windows 7 or more recent

**Formats**

Standalone, VST3 plugin

## macOS

We tested Anyma V on a variety of macOS versions and Mac computers from the last 12 years.

**Processor**

Intel 64-bit or Apple Silicon processor

**RAM**

1 GB of RAM

**Disk space**

100 MB of free disk space

**Operating system**

OS X 10.9 or more recent

**Formats**

Standalone, Audio Unit and VST3 plugins

## Linux

Although we tested Anyma V on Debian 12 on a few architectures, as well as a SBC similar to Raspberry Pi, we cannot test it extensively due to limited resources and are not able to provide support for the Linux version.

**Processor**

Intel/AMD 64-bit or ARM (32-bit armv7hf or 64-bit armv8) processor

**RAM**

1 GB of RAM

**Disk space**

100 MB of free disk space

## Operating system

Any OS with Linux 3.2.0 kernel or more recent and glibc

## Formats

Standalone, VST3 plugin

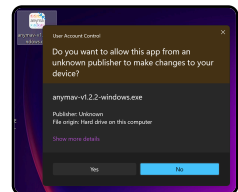
## Installation

If you haven't already, **download** the latest version of Anyma V corresponding to your operating system from our website <https://anymav.aodyo.com>.

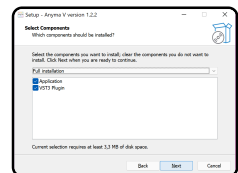
### Windows

The file you downloaded from our website should be named `anymav-win.exe` or `anymav-v1.3.0-win.exe`, unless a more recent version was disponible.

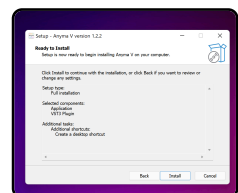
If you are sure the came from our website, **open** it and press **Yes**. If you are unsure of the origin of this file, press **No**, delete it, and download it again from our website.



After selecting the destination folder, you will be given the choice of installing the standalone application and the VST3 plugin. Other options follow after you press **Next**.



Once you are ready, press **Install** to begin installing Anyma V. After you're done, you will be offered the choice to start the standalone application directly.





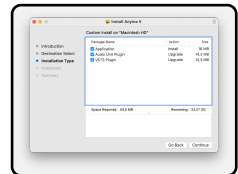
## macOS

The file you downloaded from our website should be named `anymav-mac.dmg` or `anymav-v1.3.0-mac.dmg`, unless a more recent version was disponible.

**Open** it and wait for this window to appear, then **open** the package named *Install Anyma V.pkg*.



After clicking *Continue* and selecting the destination disk, you will be given the choice of installing the standalone application, the Audio Unit and the VST3 plugins. Click *Continue* and *Install*, and enter your password or use Touch ID to confirm the installation.



Once the installation is done, you can find the standalone application in your standard `/Applications` folder.

## Linux

The file you downloaded from our website should be named `anymav-linux.zip` or `anymav-v1.3.0-linux.zip`, unless a more recent version was disponible.

**Unzip** it and you will find three pairs of files, for each of the three architectures we support (Intel/AMD 64-bit, armv7hf 32-bit, armv8 64-bit):

- `anymav-{x86|arm32|arm64}` is the standalone program: **copy** it to a folder in your `$PATH`

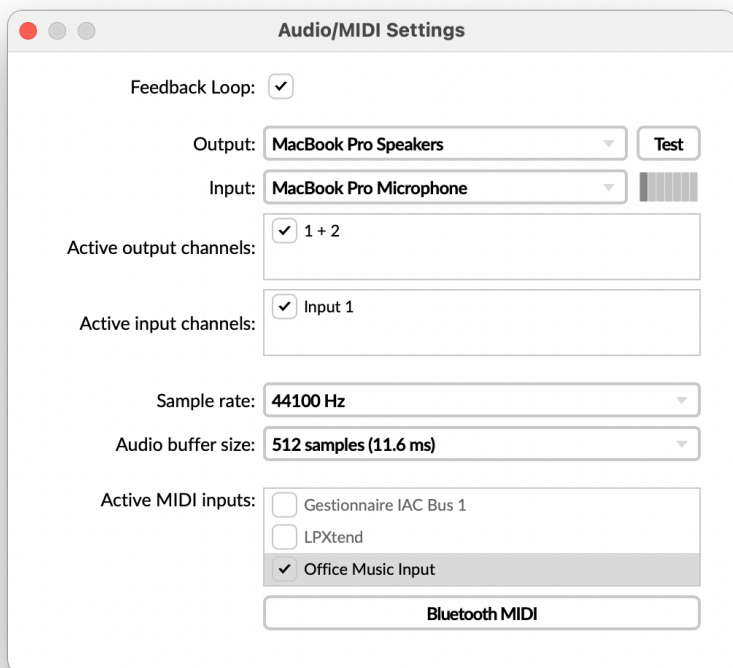
- `anymav-{x86|arm32|arm64}.vst3` is the VST3 plugin: **copy** it to your VST3 folder, such as `$HOME/.vst3`

## Setting up audio and MIDI settings

Plugins don't require you to set up audio and MIDI devices, because your DAW or plugin host has already done that for you. But to get to know Anyma V, we'll use the standalone application first.

**Open** the standalone application.


At first launch, you will see a window named *Audio/MIDI Settings* where you will need to select which audio device and which MIDI inputs to use.



**Click** the *Test* button, and if you don't hear a sound without crackles, select another *Output* device until you do. You can also uncheck the *Feedback Loop* box to be able to use the “piezo” or “effects” patches using an audio input, if you are confident there is no possibility of audio feedback loop.

To play Anyma V you will need to **connect** a MIDI keyboard or controller to your computer. It will appear in the list of *Active MIDI inputs*, and you

will need to check its box to be able to use it. If you don't have any MIDI device on hand, you can still test the software using the included virtual piano keyboard.

Once ready, **close** the *Audio/MIDI Settings* window. You will be able to summon it back later by clicking the  button in the toolbar.

## Demo mode, license and activation

At first, Anyma V will run in **demo mode**, meaning that it will be fully functional for an unlimited amount of time, except that it will periodically dysfunction for a short amount of time and show a reminder that you are using a demo. This allows anyone who is interested in the Anyma synth to try it and see if they want to **purchase a license** to run a fully unlocked software.

Purchasing a license can be done on our website. There are discounts available for owners of hardware Aodyo products, such as Sylphyo or Anyma Phi, and backers of our crowdfunding campaigns. After the purchase, you will be sent a confidential License Key, which looks like 22TZ-A40R-X4HH-4YRP-YLF3-J7HM-33V7-RLAH-Y2<sup>1</sup>.

### IMPORTANT

A License Key is the only thing needed to activate or deactivate the machines associated to it. Anyone with the key can take control of your activations, so please make sure you store the License Key in a safe place and do not share it with anyone.

---

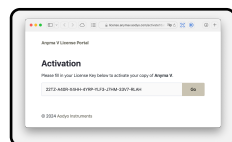
<sup>1</sup>This exact License Key does not exist! It can also have a slightly different size or format.

Owners of a license can use **Anyma V** on three computers<sup>2</sup>. To unlock the software on a computer, you must **activate** it. The activation procedure is simple and only happens once. The software itself never communicates with any server.

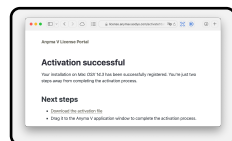
The first launch will show this *activation dialog*, which you can also recall by selecting *Activation...* in the *Help* menu. Supposing you already bought a license, if you are on an Internet-connected computer, you can activate it by clicking *Activate Online*.



Your browser will open a window to our License Portal. After checking the server is `license.anymav.aodyo.com`, enter your License Key and press *Go*.



**Download** the activation file that has been generated using the download link under the *Next steps* heading.



**Drag** the activation file onto the **Anyma V** window. When you release the mouse button, you should see a message saying *Activation successful*. You can now use **Anyma V** without limitations.



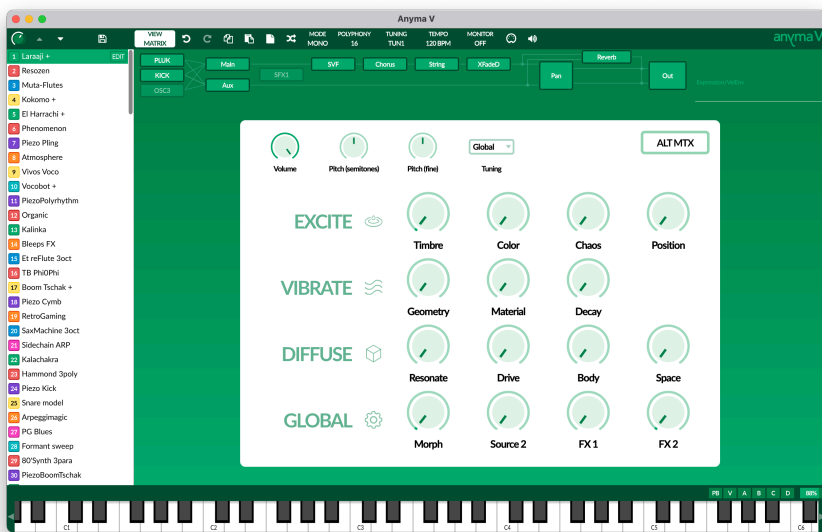
---

<sup>2</sup>To free up slots, you can deactivate a previously activated computer on the *License Portal*. However, you can only deactivate a computer once 30 days have passed since its activation.

You can also activate **Anyma V** on a computer that is not connected to the Internet. Instead of clicking *Activate Online*, you can **copy** the activation URL and transfer it to another computer (e.g., using AirDrop or a USB thumb drive), or **scan** the QR code using a mobile device. You can then download the activation file on the Internet-connected computer or mobile device, and transfer it to the computer you want to activate.

## First steps

You should now be facing this screen:



You can now **play** a few notes from your MIDI keyboard or controller, or clicking the virtual piano keyboard at the bottom, and hear a plucked string sound through your audio system. If you can't hear anything,

**adjust** the volume using the small *Master Volume knob* at the top left of the window, in the toolbar.

Try to **change** the timbre of your sound by turning any of the big knobs in the *matrix* at the center of the window. For instance, turn ☺ *Timbre* all the way up to dampen the plucked string, or to the contrary, turn ≡ *Decay* to let it ring freely. Some knobs seem to do little, while others have a more drastic effect on the sound. Don't be afraid to experiment with different combinations. You'll always be able to revert to the initial sound later.

The sound you're playing, as well as all its possible variations, is called a *patch*. Anyma V gives you access to a bank of 200 patches in a single session, and you can import and export the entire bank or individual patches.

Initially, Anyma V comes with a number of patches made by the Aodyo team, and the remaining ones are left blank and labelled (empty). Now, **switch** to another non-empty patch by clicking its name on the list on the left. Your changes on the previous patch will be forgotten, although you can still go back in time using the *Undo* command if you need to.

Some patches also allow you to directly control variations using your MIDI keyboard or controller. For instance, if your keyboard has a *modulation wheel*, try to turn it up and down while playing patch #3 Muta-Flutes, and listen to how it reacts to your gestures.

## Playing with a wind controller

Anyma V is also compatible with wind controllers, or any MIDI controller that can send *breath control* or *expression* messages (usually through CC 2 or 11).

When such a controller is detected, Anyma V automatically uses it as an *expression* signal that can modulate the volume of your sound, or the amount of excitation energy when using percussive or physical models. When you use a more typical MIDI keyboard, this expression signal is replaced by a *velocity envelope*, setup by the patch designer, that describes how volume or excitation energy evolves over time in reaction to the velocity of your notes.

This way, you can design patches that sound and feel good when played with both keyboards and wind controllers, and all patches are guaranteed to offer a basic level of expressiveness when played with a wind controller.

## Playing polyphonically

You will surely notice that you seem limited to a single note at a time, except for some patches with suffixes like *3poly* or *2p* that allow 2 or 3 simultaneous notes. This is because at first, Anyma V is in *Monophonic Input Mode*, and works exactly like a virtual Anyma Phi.

To switch to *Polyphonic Input Mode*, find the **MODE: MONO** button in the toolbar, and **click** it so that it displays **MODE: POLY**. You can now play up to 16 notes at a time.

## Playing with a MPE controller

If you are using a MPE controller, you can just click once more on the **MODE** toolbar button to switch it to *MPE Input Mode*, making it compatible with any MPE controller<sup>3</sup>.

---

<sup>3</sup>Note that you cannot configure two MPE zones, and Anyma V will only respond to the Lower Zone.



If your controller sends a *MPE Configuration Message*, it will be picked up by Anyma V and you won't even need to change its mode manually.

## Video tutorials

For a deep dive into the synth engine, we recommend watching the following video tutorials. They have been made for the Anyma Phi, but they mostly focus on the patch editor, which is almost identical in Anyma V.

VIDEO

[youtu.be/eplA-\\_LrWTA](https://youtu.be/eplA-_LrWTA)


► **Anyma V Review by Loopop**

VIDEO

[youtu.be/eplA-\\_LrWTA](https://youtu.be/eplA-_LrWTA)

► **Tout sur le synthétiseur Anyma Phi by DRJ (in French)**

## Adjusting mappings

To adjust Anyma V to your controller, press the  button in the toolbar. From there, you will be able to select:

- what the MPE Pressure and Slide controls influence in the synth,
- the pitch-bend range
- which MIDI CCs change the *Expression* input
- which MIDI CCs change the *Control A/B/C/D* inputs that are used by the synth in various ways



# The matrix

At this point, you can already play and explore all the included patches. In this chapter, we will learn the logic behind these variations we've experimented with.

The starting point of **Anyma V** is its **matrix**, which provides a quick and easy way to tweak your sounds by following a physical metaphor, with 15 parameters governing the four stages of sound generation:

**Excite** 🎵 *Timbre, Color, Chaos, and Position*

How the sound is initiated or sustained with external energy. In the real world, it could be a bow or a plectrum.

**Vibrate** 🌀 *Geometry, Material, and Decay*

How the excitation energy propagates into a medium, such as a string or a tube, giving it its timbre and response.

**Diffuse** 📦 *Resonate, Drive, Body, and Space*

How the sound is shaped by the overarching resonating structure, such as the body of a violin.

**Global** ⚙️ *Morph, Source 2, FX 1, and FX 2*

How the sound is blended with another source or effects (which depend on the patch).

## The matrix is the interface to your patch

If you've ever dealt with synthesizers before, you might notice that this matrix looks fairly different from the world of oscillators and filters you're used to. You might even wonder whether you can get any


electronic-sounding stuff out of such a machine. And the answer is: yes you can, and even though it doesn't look like it, there may well be oscillators and filters behind this matrix.

In fact, it doesn't matter whether the sound you hear actually resembles that of an acoustic instrument or not, because you could always apply this metaphor to it. With an acid bass, for instance, controls more related to the attack of the sound will likely be at the *Excite* stage, while the *Vibrate* controls will probably be more about the overall timbre.

This way, the matrix offers a sort of *common tongue* spoken by all Anyma users, allowing you to more easily tweak any sound you can load into Anyma V in ways that have been carefully designed by the creator of this sound.

In other words, the matrix is the *interface* of a sound. It's the command panel its designer has left you to play with their sound and make it go to unexpected places. Once you're confident enough to start designing your own sounds<sup>1</sup>, you'll find the matrix an ideal place for controlling complex variations in just a few turns of a button, which is especially useful in live situations.

## Morphing

Hidden at the beginning of the  *Global* line is a very special parameter, *Morph*, that allows you to gradually turn the current matrix into its *alternate* state. What this means is that you can actually have two versions of the matrix in a patch, which could lead to two drastically different sounds, and cross-fade between both using *Morph*.

---

<sup>1</sup>Which hopefully you'll have a blast doing very soon!

In some sounds, *Morph* can even move according to an external control<sup>2</sup>, making it effortless to span a wide range of different timbres.

Thus far, you've interacted with the matrix in its *normal* state, the state it's in when *Morph* is set at 0%. When *Morph* is set at 100%, the patch only uses the *alternate* state of the matrix.

To access this *alternate* state, press the **ALT MTX**. In the *alternate* state, you will be able to switch to any line of the matrix and modify any parameter, but your changes will not impact the matrix in its *normal* state.

Remember that switching to the *alternate* state doesn't guarantee you will hear the changes you're making, as you will also need to set *Morph* at 100% in order to make sure it only uses the *alternate* state of the matrix.

---

<sup>2</sup>In most sounds that came with Anyma V, you can move *Morph* using the modulation wheel of your MIDI keyboard, or the equivalent on your controller.



# The anatomy of a patch

There are situations where you might want to dive a bit deeper into the essence of a sound. Maybe you're not satisfied with the controls offered in the matrix. Or you'd like to create a new patch entirely from scratch. Or maybe you simply prefer dealing with a synth in terms of its basic elements like oscillators and filters, and do not wish to spend much time manipulating the matrix.

In this chapter, you will find a gentle introduction to the synth engine of **Anyma V**. If you're already familiar with synthesizers, you might be tempted to skip all this. However, we recommend that you don't because this chapter also describes the specific way **Anyma V** works compared to other synths.

Beyond the matrix lies a powerful semi-modular synth engine, with oscillators covering a wide range of synthesis techniques, effects like filters and delays, and modulators such as envelope generators and LFOs. By connecting those together, you can create any kind of sound, and control it any way you want.

You can switch between the matrix and the editor views by clicking the **VIEW** button in the toolbar.

## Introduction

In the acoustic world, a musical instrument is the physical object that both:

- produces the sound waves you can hear (e.g., the vibrating reed controlling the air flow in a saxophone, and the tube where the resulting sound waves travel),
- and offers you ways to control how the sound waves are produced (e.g., the keys of the saxophone that allow you to control its pitch).

In the electronic world, these responsibilities are usually split into two distinct parts.

The former part is called a **synthesizer**, which produces an electric current that is later turned into sound waves by your sound system (e.g., headphones, or an amp).

The latter part is called a **controller**, which uses sensors that respond to your gestures as a performer in order to produce a stream of data (usually in the MIDI format) that tells the synthesizer what kind of sound to produce (e.g., *play a C note very loudly*).

Anyma V is such a synthesizer (albeit embedded in a computer program), so its responsibility is to understand what a controller sends it, and to turn it into sound, so that the whole formed by the controller-synthesizer couple feels like an expressive and powerful instrument. Now, where does the sound come from?

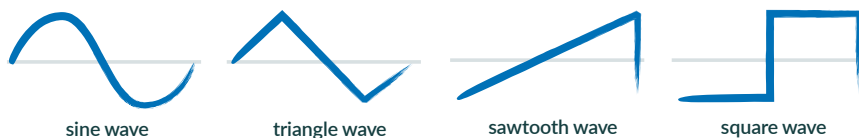


## Oscillators

The main building blocks of a synthesizer are **oscillators**: elements that create sound out of nothing.

The oscillators in the first synthesizers were electronic circuits which produced repeating patterns of electric signals that oscillate (hence the name) so quickly that you can hear a distinct pitch in the resulting sound. For example, if the oscillator is tuned to a *frequency* of 440 oscillations per second (or 440 Hz), you will hear an A note, independently of the shape of the electric signal (the *waveform*).

There were different kinds of waveforms an oscillator could produce, and all had different timbres, which could be used to approximate different instruments, or to create various kinds of sounds.



For instance, a sine wave oscillator produces a pure but dull tone, while a square wave sounds much harsher and has a very rich harmonic content.

Nowadays, oscillators can be much more sophisticated, and in addition to the traditional single-waveform oscillators, you will encounter many other ones that all sound different and are based on different synthesis techniques. Here are a few examples:

- *Virtual analog* techniques strive to reproduce the warm and imperfect sound of the oscillators in analog synthesizers.
- *FM* (frequency modulation) is the use of a simpler oscillator to modulate the frequency of another oscillator, resulting in a widely different timbre from the original.
- *Physical modelling* techniques aim at simulating the physical structures of acoustic instruments (winds, strings, etc.) so that the resulting sounds behave like the instruments they're modelled after.

In Anyma V, you get up to three oscillators that you can mix together or have played separately, depending on what kind of sound you want to achieve. Some are very simple and produce a single tone, others are much more complex and can produce multiple and very rich tones on their own. Once you'll be familiar with the palette of oscillators in the synth, you'll know which one to use to get any kind of sound.

Each oscillator is a **module** that presents several **parameters**; i.e., controls that you can change to customize how the oscillator produces its sound. Some parameters are related to pitch (e.g., you can set an oscillator to an octave higher, or detune it a little), some are related to timbre (e.g., some sounds can be dampened or brightened), and some are related to even other aspects. When setting up an oscillator, you just select one by changing its *type*, and tweak its parameters so that it sounds right to you.

But it is often necessary to fine-tune or even drastically change the sound coming out of one or several oscillators to get what you really want. That's what effects are for.

## Effects

**Effects** are modules that take some sound and transform it in a way or another. This is a rather vague definition, because there are many different types of effects. Let's see a few examples.

**Amplifier** effects simply change the volume at which the sound is heard. When turning the volume down, you simply hear a quieter version of what's put into the effect, but when turning the volume up, some amplifiers *distort* the resulting sound, much like guitar amplifiers turn the clean sound of an electric guitar into a warm, grainy sound.

In traditional analog synthesizers, the most commonly used effects were **filters**, which change the timbre of the sound. Low-pass filters can make the sound duller, by cutting high frequencies. Conversely, high-pass filter can make the sound thinner by cutting low frequencies. There are other kinds of filter *modes* that perform various different operations on the frequency contents of a sound. Some filters can also add *resonance* by amplifying some frequencies, giving a resonant quality to the resulting sound.

Some effects are based on adding delayed versions of the input sound into itself, which can have drastically different results depending on the length of the delay and how the signals are mixed together. When the delay is long, such as in **delays** and **reverbs**, it can sound like echo or simulate the reverberation of a room or hall. When the delay is much shorter, it can sound like there are multiple similar sounds at the same time (**chorus**), like in a choir.

Like oscillators, you can customize an effect module by tweaking its parameters. And as with many effects in the wild, most effects offer you

two special *Send* and *Dry/wet* parameters. *Send* allows you to choose how loud is the input sound sent into the effect, and *Dry/wet* allows you to mix the *dry* sound (untouched by the effect) and the *wet* sound (the result of applying the effect).

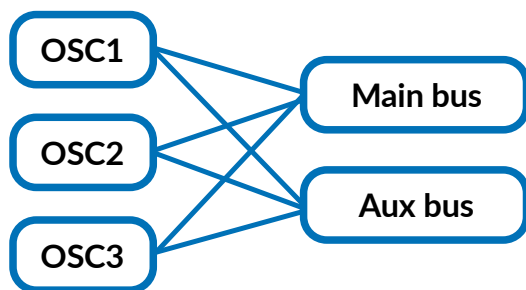
In Anyma V, you can use up to five different effect modules, with an additional final *Reverb* effect that applies to the final sound as a whole. Now, these effects don't do anything in isolation, as they need to take sound from somewhere. So how are oscillators and effects interconnected?

## Audio signal path

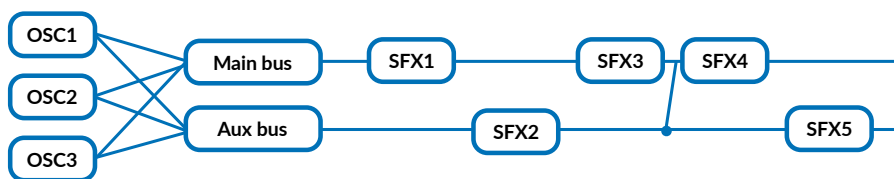
The way sound flows from oscillators to effects to the synth's output will be referred to as the **audio signal path**. There are many different philosophies for determining the signal path: modular synthesizers allow you to plug oscillators and effect modules freely, while more traditional synthesizers can sometimes have a predetermined and fixed signal path. Having a predetermined signal path is more restrictive, but it makes it easier to design sounds, because the modules are placed in predictable places and no time is spent connecting all modules together.

Anyma V can be seen as a *semi-modular* synth: the signal path is predetermined as well, but there are a few degrees of freedom as to how the signal flows, so that it is generally possible to achieve exactly what you want.

Each of the three oscillators (OSC1, OSC2, OSC3) outputs its sound into two **buses**: the **Main bus** and the **Aux bus**. You can freely choose how much of each oscillator is in either bus.

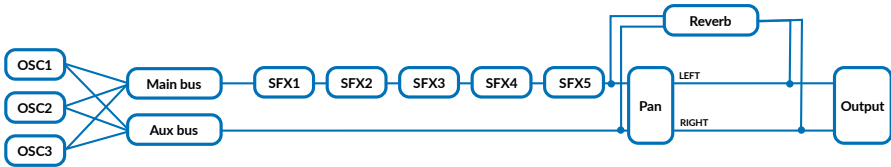


Now, each of the five effects (SFX1, SFX2, SFX3, SFX4, SFX5) are at a fixed place on the signal path, but they can be placed on either bus, allowing you to have two different ways of applying effects to different mixes of the oscillator outputs. For instance, in the example below, SFX1, SFX3, and SFX4 stay on the Main bus, while SFX2 and SFX5 are on the Aux bus. Also note that SFX4 is a specific kind of effect, a **mixer**, whose task is to take the sound of both buses at that point, and to mix them in a certain way so that the resulting sound can only be heard on the Main bus, leaving the Aux bus untouched.



After all the effects have been applied, you can control how much of each bus you want in the left or right channel of the stereo output (**panning**, or **pan**). At the end of the chain, there's also a mono **reverb** that takes a mix of the Main and Aux bus, and adds its reverberation

into the synth's output. Here's an example with all the five effects in the Main bus:



This is how the audio signal path works in Anyma V. If it looks a bit complicated to you, just remember that you don't need to mess with it when starting to create sounds.

By default, all three oscillators output on both buses, the Aux bus is muted (only the Main bus works), and all five effects are placed on the Main bus. If the signal path looks a bit restricted to you, please keep in mind that using the *dry/wet* controls of most effects, you could still achieve much more complex mixes.

## Mappings

We're now at a point where all the sound-producing components of the synth are in place. You can set them up however you like and get a wide variety of sounds out of this, but there's a crucial bit missing: how to make everything move and react to what you play?

This is the job of **mappings**. A mapping describes how a *control signal* (the note you're playing, the modulation wheel on your MIDI keyboard, a control in the matrix, etc.) influences a parameter of any module in the synth<sup>1</sup>.

<sup>1</sup>Even another mapping! But let's not get carried away too soon.

For instance, you might want to make the sound darker or brighter depending on how hard you press a key on your MIDI keyboard. Or you might want to make it so that pressing the modulation wheel drenches your sound in a lush reverb. To do so, you can set up a mapping, choose a control signal (the *source*), which parameter of a module it influences (the *destination*), and the amount by which it will move that parameter (the *amount*).

Each mapping can also be customized so as to only take into account some part of the source signal (e.g., the first half of your modulation wheel), to apply a non-linear *curve* to the relationship, to *smooth* out the signal over time, or even to make it depend on another *sidechain* source to build complex relationships and interdependencies.

Anyma V offers up to 32 mappings in a single patch, allowing you to turn a dull sound into a lively and expressive one that you can control in many different ways.

But in musical instruments, not all the work is done by the performer. Sometimes, things oscillate by themselves, act seemingly randomly, or take a bit of time to appear. To be able to reproduce such phenomena, we would need to create new control signals which can be used as sources in mappings, but that don't really originate from the controller. Modulators are the modules that allow this.

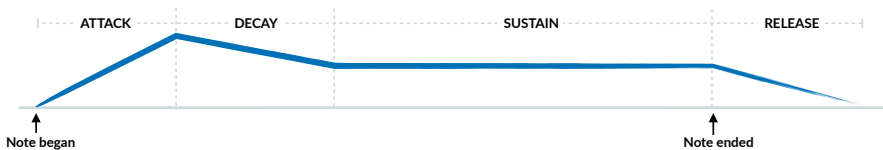
## Modulators

**Modulators** are internal sources of control signals, independently from the external signals coming from your controller and from the matrix controls. They offer you various ways to make your sound livelier, and sometimes can even have a drastic effect on your timbre. There are

many different kinds of modulators, but let's focus on the two most well-known ones.

Traditional synthesizers are often controlled using a keyboard, but compared to a wind controller such as the *Sylphyo*, a keyboard offers relatively few control signals to play with: they're even sometimes limited to a pitch (the note you're playing) and a velocity (the intensity at which you hit the note on the keyboard). If that were all there is, synths would sound very bland, as in an instrument the timbre usually doesn't stay exactly the same from the beginning to the end of a note. Some notes can start slowly, then fade a little, and some even take some time to disappear, or their timbre can change over time. The timing is given by a modulator called **envelope** generator, which creates a control signal that evolves in time, from the start to the end of a note. Typical envelopes, called *ADSR*, work in four successive phases<sup>2</sup>:

- When a note starts, the *attack* phase begins, and the control signal increases from 0% to 100%.
- Once the signal reaches 100%, the *decay* phase begins, and the control signal decreases a bit to a level called the *sustain value*.
- Once this value is reached comes the *sustain* phase, and the control signal stays still.
- When the note ends, the *release* phase begins and the value decreases to zero.



<sup>2</sup>But keep in mind that there are many variations of this principle out there.



Another modulator typically used by traditional synthesizers is the **LFO** (low-frequency oscillator). As the name says, it creates an oscillating control signal, much like a simple oscillator operates, but at a much lower frequency (or *rate*), with a chosen waveform. Using a mapping, you can then make any parameter of your synth oscillate. Using another mapping, you could also change, for instance, the rate of the LFO depending on another control signal. You could also, for instance, make the oscillation appear only when the modulation wheel of your keyboard is up, in a gradual way.

In addition to envelopes and LFOs, Anyma V offers a wide range of other modulators that can filter, shape, smooth, or mix other control signals in various ways, making almost any complex modulation possible. Anyma V allows up to 16 modulators in a patch.

## Velocity Envelope and expression

In addition to the 16 freely choosable modulators, your patch includes a *Velocity Envelope* that will generate an envelope signal depending on the velocity of the last played note.

This envelope offers two sets of attack/decay/sustain/release parameters, one when the velocity is at its highest point, and another one when it is at its lowest point. The actual envelope will use a mix of these two extreme settings depending on the exact velocity of the last note.

This envelope signal will be used when you play a MIDI keyboard to generate a more complex *expression* signal that will be used by the oscillators. Most oscillators use the expression signal to control their output level, but percussive or resonant oscillators, as well as physical models,

use it as a level of energy that will make the resulting sound louder or softer, together with complex timbre variations.

When you're playing a wind controller, or using a controller that sends *breath control* or *expression* messages, the synth will use that directly as its *expression* signal, bypassing the velocity envelope altogether.

This allows you to design patches that will sound good when played with a keyboard and a wind controller, without needing to setup complex mappings from the ground up.

## Connecting the synth to the matrix

What's been presented so far is much more similar to a typical synthesizer than what the matrix makes it look like. In fact, the matrix is nothing more than a collection of control sources that you can map to any destination parameter. It's up to you to decide how each control in the matrix will act on your patch.

For instance, if you're using resonators or resonating oscillators, it might be wise to map their decay or damping to  $\approx$  *Decay* in the matrix.

⚙ *Source 2*, *FX1*, and *FX2* (in the *Global* line), can be used to control the level of another oscillator, or the dry/wet level of strategically chosen effects.

You can setup mappings so that  $\boxtimes$  *Space* increases the time and level of the final reverb.

You can also leave some matrix controls unassigned.

There's no right or wrong choice, as long as the patch makes sense to you and those whom you share it with.

## Patch structure

At this point, we've seen most of the components of a patch:

- 3 oscillators (*OSC1*, *OSC2*, *OSC3*) that make sound out of nothing,
- the *Main* and *Aux* buses that connect the oscillators and inputs to effects,
- 5 effects (*SFX1* to *SFX5*) plus a final *Reverb* that transform the sound on the bus,
- 16 modulators (*MOD1* to *MOD16*) plus the velocity envelope (*VelEnv*) that generate internal control signals,
- 32 mappings (*MAP1* to *MAP32*) that describe how internal and external control signals act upon the patch in real time,
- and the matrix plus its alternate version.

GENERAL							
Patch <i>Tuning, volume, color...</i>	VelEnv <i>Velocity envelope</i>	Main <i>Mix for Main bus</i>	Aux <i>Mix for Aux bus</i>	Verb <i>Final reverb</i>		Mtx <i>Matrix controls</i>	MtxAlt <i>Alternate matrix controls</i>
OSCILLATORS			EFFECTS				
OSC1	OSC2	OSC3	FX1	FX2	FX3	FX4	FX5
MODULATORS							
MOD1	...						MOD16
MAPPINGS							
MAP1	...						MAP32

Each of these things we just described is a **module** that contains a bunch of **parameters** that completely describe it. In addition, there is a special *Patch* module that contains some general parameters, such as global tuning or color.

In fact, we can see that a **patch** is nothing more than a particular configuration of the synth at a given point, the set of all the parameters of

all its modules<sup>3</sup>. It completely describes a sound and how it reacts to your controller and to the matrix.

## The bank

As with most synthesizers, you can switch between different patches, and Anyma V offers up to 200 different patches, making up a **bank**.

You can only have a single bank at a time open in Anyma V, but you can export it to a file, and import any bank created from elsewhere. You can also import and export individual patches, so that you can share your creations and use those of others.

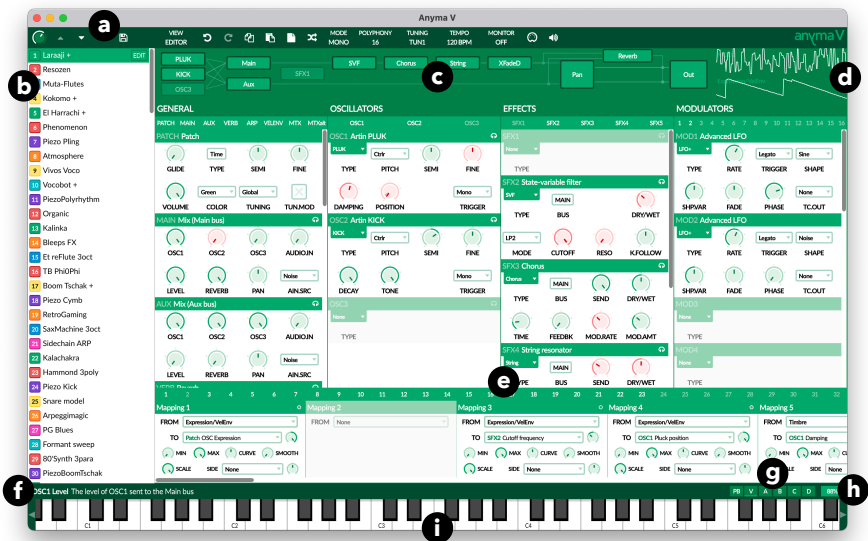
---

<sup>3</sup>Up to 960 in a very stuffed patch!

# Editing patches

By centering on its matrix, the **Anyma V** does not make it obvious that you can easily edit your patches there, and might even seem to be shielding you from the reality of the synth engine behind it. This is not only to make tweaking more immediate, but also to allow novice users to freely play existing sounds and leave them a wide field of sonic exploration without having to learn all the nuts and bolts.

But you will see that it's quite possible to modify existing patches in depth, or even to create entire patches from scratch.



**a Toolbar**


Offers important actions and settings.

Hover the mouse over a control, text or icon to see what it is about.



**b Patch list**

Displays the current bank of patches.

Click on a patch to select and edit it.

After editing a patch, make sure you **save** it first using Ctrl/Cmd+S, or the  toolbar button, otherwise your modifications will be lost. Right-click on the selected patch to rename or clear it.

Drag the selected patch to your desktop to export it as a file.

You can also copy and paste patches by using Ctrl/Cmd+Shift+C and Ctrl/Cmd+Shift+V, or the corresponding  and  toolbar buttons.

You can export the bank to a file using the relevant command in the *File* menu.

**c Audio signal view**

Shows the relationships between oscillators, effects, buses, and the other elements in the audio signal chain.

Click a module to focus on it.

**d Scope views**

Show the evolution of two control signals.

By default, the top scope displays the last selected modulator, and the bottom scope displays the expression / velocity envelope signal.

Right-click a scope to select which control signal to display.

#### **e Patch view**

Arranges all the modules in the patch by category, and allows you to modify any parameter by clicking or dragging it.

You can undo the last operation by using Ctrl/Cmd+Z, and redo it by using Ctrl/Cmd+Shift+Z.

Right-click a parameter to reset its value to the default one, go to its related mappings, or create a new mapping.

Click the title bar of a module to focus on it.

Drag the title bar of an oscillator, effect, modulator, or mapping module to rearrange it.

Right-click the title bar of a modulator module to reset it, go to its related mappings, or create a new mapping.

Drop a patch file in this area to import it, replacing the current patch.

Drop a bank file in this area to import it, replacing the entire bank.

#### **f Contextual help**

Offers details about the element under the mouse pointer.

#### **g External control meters**

Display the current received MIDI pitch-bend, expression or velocity envelope, and Control A/B/C/D.

#### **h Anyma Phi patch load meter**

Display the amount of resources that would be used by the current

patch if it were loaded into an Anyma Phi hardware synth. This is only useful if you plan to use the patch in the hardware synth.

**i Virtual piano keyboard**

Click a key to play the corresponding note.

You can also use this zone to monitor the notes currently played using your MIDI controller.



# Tuning and microtonality

You can tune **Anyma V** to any reference frequency or tuning system, such as just intonation, historic or non-Western tunings, or those of modern microtonal and xenharmonic music. It also integrates seamlessly with MTS-ESP to allow you to dynamically control tuning in a global fashion across your plugins.

In this chapter, you will learn how to interact with tunings, how they can be transferred in and out of **Anyma V**, as well as how you can load some included preset tunings.

Up to 8 tunings can be stored, from *TUN1* to *TUN8*. Each tuning has a name and a definition, and can be created directly on the **Anyma V** or can be imported from other systems via the MIDI Tuning Standard, or from files: Scala SCL/KBM pairs are supported, as well as MTS SysExes. By default, the global tuning is set to the standard 12-tone equal temperament (12-TET, or 12edo), with A4 sounding at 440 Hz. You can select a tuning from *TUN1* to *TUN8* to act as the global tuning, which will be used by the synth instead of the 12-TET. Each patch can also follow a specific tuning (with the TUNING parameter in the Patch module), but by default the global tuning will be followed.

When selecting or changing a specific tuning in the **TUNING** toolbar item, you will be able to listen to its effect by playing your MIDI keyboard or controller, but only if it is selected as the global tuning or as the patch tuning.

## Importing tunings

If you already have your tunings ready, either in the Scala (.scl/.kbn) or MTS (.syx) format, you can import them to Anyma V by selecting the *Import SCL/KBN/both/SYX* menu item from the **TUNING** toolbar item.

If you need to edit or convert your tuning, you can use a tool such as *Scale Workshop* to set it up with the correct format for import.

## Preset tunings

The following preset tunings are offered in the **TUNING** toolbar item. When not indicated otherwise, the tuning has 12 tones, starts with C4, and has C4 as the reference note as well.

### None (12edo)

This is the standard 12-tone equal temperament (12-TET, or 12edo), with A4 sounding at 440 Hz.

### 12edo with reference tone (A4)...

This is the standard 12-TET tuning, but you will be offered the choice of the reference frequency (in Hz) for the A4 note.

## Non-Western tunings

### Pelog

A heptatonic scale used in Javanese gamelan.

### Slendro

A pentatonic scale used in Javanese gamelan.

### Rāgas (Bāgashri, Bhairavi, Kafi, Todi, Yaman)

A series of heptatonic tunings that can be used in Hindustani rāgas.

### 22 Shruti

A tuning system for Indian music composed of 22 shruti or intervals.

## **Hirajōshi**

A pentatonic Japanese scale used for the tuning of the koto.

## Well temperaments

The following are 12-tone unequal temperaments commonly used in baroque music.

### **Bach-Kellner**

A temperament with 7 perfect fifths.

### **Werckmeister III**

The most well-known unequal temperament described by Andreas Werckmeister in *Musicalische Temperatur* (1691).

### **Kirnberger III**

A temperament with a single perfect third.

### **Vallotti/Young**

A temperament described by Francesco Antonio Vallotti, later transposed by Young.

### **Neidhardt III**

A temperament described by Johann Georg Neidhardt in 1724, which he famously recommended "*für eine große Stadt*".

## Meantone temperament

### **1/4 comma meantone**

The most commonly used meantone temperament in the 16th and 17th centuries.

### **1/3 comma meantone (Salinas)**

A temperament first described by Francisco de Salinas in *De musica libri septem* (1577).

### **1/5 comma meantone (Verheijen)**

A temperament attributed to Abraham Verheijen and Michelangelo Rossi.

### **1/6 comma (tritonic)**

A meantone temperament making the tritone pure.

### **2/7 comma (Zarlino)**

A temperament described by Gioseffo Zarlino in *Le istituzioni harmoniche* (1558).

## Just intonation

### **Pythagorean**

A tuning system, attributed to Pythagoras in the sixth century BC, based on the ratio  $3/2$ .

### **Zarlino**

A tuning, also called Zarlino's system, based on the combination of pure intervals.

### **Harmonic scale**

A 12-tone scale based on the harmonic series proposed by Wendy Carlos.

### **7-limit tritone**

A basic just intonation tuning with a septimal tritone.

### **Fokker**

Adriaan D. Fokker's 7-limit 12-tone just scale.

### **SuperJust**

Wendy Carlos' SuperJust intonation scale.

### **Centaur**

A 12-tone 7-limit mesotonal scale proposed by Kraig Grady.

### **Centaura**

A 12-tone 11-limit scale proposed by Kraig Grady.

### **Centaura subh.**

The original subharmonic version of Centaura.

### **Natural Harm. 16**

The 16-tone natural harmonic scale proposed by Bruce Kanzenmeyer, directly generated from the relationships of the harmonic overtone series.

## Non-octave tunings

The following tunings do not include octaves.

### **Alpha scale**

The Alpha scale proposed by Wendy Carlos, with 78.0 cents per step.

### **Beta scale**

The Beta scale proposed by Wendy Carlos, with 63.8 cents per step.

### **Gamma scale**

The Gamma scale proposed by Wendy Carlos, with 35.1 cents per step.

### **Bohlen-Pierce equal**

The equal-tempered form of the Bohlen-Pierce scale.

### **Bohlen-Pierce just**

The just form of the Bohlen-Pierce scale.

## Equal temperament

### **Pentatonic**

A tuning system where the octave is divided in 5 equal steps (5edo) of 240 cents each.

### **Whole-tone**

A tuning system where the octave is divided in 6 equal steps (6edo) of 200 cents each.

### **Heptatonic**

A tuning system where the octave is divided in 7 equal steps (7edo) of approx. 171.4 cents each.

### **19edo**

A tuning system where the octave is divided in 19 equal steps of approx. 63.2 cents each.

### **31edo**

A tuning system where the octave is divided in 31 equal steps of approx. 38.7 cents each.

### **53edo**

A tuning system where the octave is divided in 53 equal steps of approx. 22.6 cents each.

## Using MTS-ESP

Anyma V can also act as a client for the MTS-ESP Suite, allowing you to dynamically control the tuning of the synth from a master tuning system. When MTS-ESP is enabled from the **TUNING** toolbar item, the synth will follow the tuning sent by the master system, disabling the local tunings. When a connection is established with the master, the toolbar item will be highlighted.

# Release notes

## v1.3.0

- Added new *Bass Brass*, *Flute*, *Clarinet* and *Cylsax* physical models.
- Added new *Bagpipe*, *Corroboree*, and *Overdrive guitar* oscillators based on Sylphyo presets.
- Added user interface scaling options.
- Separated MIDI mode settings into two *Mono/Poly* and *Legacy/MPE* switches, opening the possibility of mono-legato playing using a MPE controller.
- Replaced the *Patch TUN.MOD* parameter with a new *Polyphony mode* parameter, allowing you to either follow the global setting, or to force poly or mono mode.
- Added MTS-ESP client support for dynamic tuning.
- Singled out *Morph* in matrix mode presentation.
- Added a *Slow Envelope* modulator.
- Added a *DETUNE* parameter to the *Artin X3* oscillator.
- Added *RT.FINE* parameter to the *Artin FM* oscillator.
- Increased the range of the *Drawbar organ* by one upper octave.
- Fixed a bug with *Delay sync* in polyphonic mode.
- Made the velocity envelope hard reset when a new note is played.
- Added a *0 st* option for the pitch-bend range.
- Added 126 new patches to the factory bank (thanks to Dan Maurer for donating much of them).

## **v1.2.2**

- Fixed a bug where Glide would not work properly in MIDI Mono mode.
- Fixed a bug where exporting a bank would fail and create an empty file.
- Fixed a bug where importing Anyma Phi patches would result in wrong Audio In levels.
- Added an “Open activation file...” button in the Activation dialog box.

## **v1.2.1**

- Fixed a bug where no sound would be heard on some installations.
- Fixed a bug where activation would sometimes fail on Windows. Deactivating from the software and dragging the activation file again might be necessary.
- Fixed a bug where the plugin would sometimes crash on macOS and Windows.
- Fixed a bug where the demo version would randomly ignore all played notes after the demo reminder screen appeared.
- Fixed a crash that could occur when reloading a plugin instance after having released all other instances.
- Added a “Copy Diagnostics Info” button in the About box.
- Added an error number for activation issues.

## **v1.2.0**

Initial version. The version number follows the current version of the Anyma Phi firmware, with the intention of keeping the major and minor version numbers (1.2) in sync.



# Complementary information

Please take note of the following important information before you begin to use **Anyma V**.

## Credits and licenses

### **Original design of the Anyma Phi**

Laurent Pouillard, Romain Bricout, Jonathan Aceituno, Ludovic Potier

### **Sound design**

Romain Bricout, Laurent Pouillard (special thanks to Dan Maurer for donating patches)

### **Software design and development, user manual**

Jonathan Aceituno

### **Marketing and social media**

Léo Barbet, Matthias Couche, Ludovic Potier

### **Sales**

Matthias Couche

### **Testing and support**

Léo Barbet, Matthias Couche

Portions of the synth engine use code adapted from *Mutable Instruments' Eurorack modules* made by Emilie Gillet, as well as modifications from the *Parasites hacks* by Matthias Puech, available under the same license.

Copyright (c) 2012-2015 Emilie Gillet

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The *Snare drum* resonator is derived from the physical model by Schmalfuß et al. at the *Center for Haptic Audio Interaction Research (CHAIR)*.

Schmalfuß, Neupert and Kessler (2020). **Efficient Snare Drum Model for Acoustic Interfaces with Piezoelectric Sensors.**

*In Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20), Vienna: Vienna University of Music and Performing Arts.*

Portions of the synth engine use a fast power-of-two approximation code by Paul Mineiro.

Copyright (C) 2011 Paul Mineiro All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Paul Mineiro nor the names of other contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER

OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contact: Paul Mineiro [paul@mineiro.com](mailto:paul@mineiro.com)

Portions of the software use the Heatshrink data compression library.

Copyright (C) 2013-2015, Scott Vokes [vokes.s@gmail.com](mailto:vokes.s@gmail.com)  
All rights reserved.

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA

OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Portions of the software use BarelyML.

Copyright (c) 2023 MNSignalProcessing

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN

AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Portions of the software use a modified version of Gin's stack blur algorithm.

Copyright (c) 2018, Roland Rabien All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY

AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Portions of the software use *libsamplerate*.

Copyright (c) 2012-2016, Erik de Castro Lopo [erikd@mega-nerd.com](mailto:erikd@mega-nerd.com) All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Portions of the software use the *Surge Synth Team Tuning Library*.

Copyright 2019-2020, Paul Walker

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:



The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Our software uses fonts Lato and Font Awesome, licensed under the SIL Open Font License, as well as fontaudio (icons: CC BY 4.0, fonts: SIL OFL 1.1, code: MIT license).

Copyright (c) 2001, Jason Kottke ([jason@kottke.org](mailto:jason@kottke.org)), with Reserved Font Name Silkscreen.

Copyright (c) 2010-2020, tyPoland Lukasz Dziedzic ([lato-fonts.com](http://lato-fonts.com)), with Reserved Font Name Lato.

Copyright (c) 2016, Dave Gandy ([fontawesome.com](http://fontawesome.com)), with Reserved Font Name Font Awesome.

Copyright (c) 2019 Michelangelo Nottoli (<https://github.com/fefanto>), with Reserved Font Name fontaudio.

## Trademarks

All the trademarks cited in this documentation are only used here for descriptive purposes. They remain subject to legal regulations and are owned by their respective property holders.

## Disclaimer

This documentation represents the current state of the product, however it may evolve. We do everything we can to provide our users with high-quality products and documentation. However, Aodyo does not guarantee that this documentation exactly reflects the state of the product. Aodyo is not liable for data loss or damage resulting from the use of this product and/or its documentation.

# Appendix: Module reference

This section details all modules and parameters that you can encounter in the synth engine of **Anyma V**.

---

## Module name

---

Patch PATCH

Velocity envelope VELENV

Mix (Main bus) MAIN

Mix (Aux bus) AUX

Reverb VERB

Arpeggiator ARP

Matrix MTX

Matrix (alt.) MTXalt

Oscillators OSC1 to OSC3

Effects SFX1 to SFX5

Modulators MOD1 to MOD16

Mappings MAP1 to MAP32

---

## General

Module name	Label
<b>Patch</b> Global patch parameters	PATCH
<b>Velocity envelope</b> The envelope used to create the Expression signal from MIDI velocity, when expression or breath control are absent	VELENV
<b>Mix (Main bus)</b> The mixer for the Main bus	MAIN
<b>Mix (Aux bus)</b> The mixer for the Aux bus	AUX
<b>Reverb</b> The reverb effect at the end of the signal chain	VERB
<b>Arpeggiator</b> Turns polyphonic note input into arpeggios	ARP
<b>Matrix</b> Macros presented in the matrix	MTX
<b>Matrix (alt.)</b> Alternate version of the matrix (for morphing)	MTXalt

## Patch (PATCH)

### Global patch parameters.

#### **Volume** VOLUME

The volume of the entire patch.

0 to 100%

#### **Patch color** COLOR

The color of the front panel lights and editor when the patch is selected.

- *Grey*
- *Red*
- *Orange*
- *Yellow*
- *Green*
- *Teal*
- *Blue*
- *Violet*
- *Pink*

#### **Pitch (semitones)** SEMI

The global pitch correction on the controller.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The global pitch correction on the controller.

-100 to +100 cents

#### **Glide time** GLIDE

The time taken to smoothly transition between two notes.

0 to 5 seconds

**Glide type** TYPE

The kind of smooth transition between notes to use.

- *Constant-rate*

Synth-like glide, where the time to get to another note is proportional to how far away it is.

- *Constant-time*

Realistic glide, where the time to get to another note is constant.

**Tuning** TUNING

The tuning to use for this patch.

**Polyphony mode** MODE

The polyphony mode to use for this patch.

- *Global*

Follow the global polyphony mode in the settings.

- *Polyphonic*

Allow multiple notes to be played simultaneously.

- *Monophonic*

Only allow one note to be played at a time.

## Velocity envelope (VELENV)

**The envelope used to create the Expression signal from MIDI velocity, when expression or breath control are absent.**

This module provides two sets of ADSR parameters, one when the MIDI velocity is at its maximum value (127), and the other when the velocity is at its minimum value (1). The resulting parameters will be between these two extremes, and determined by interpolating the parameters by the velocity of the last note.

### **Attack time (MaxVel) ATTACK**

The time needed to go from 0 to 1 in the attack phase.

0 to 8 seconds

### **Decay time (MaxVel) DECAY**

The time needed to go from 1 to the sustain level in the decay phase.

0 to 8 seconds

### **Sustain level (MaxVel) SUSTAIN**

The output level during the sustain phase.

0 to 100%

### **Release time (MaxVel) RELEASE**

The time needed to go from the sustain level to 0 in the release phase.

0 to 8 seconds

### **Attack time (MinVel) ATTACK**

The time needed to go from 0 to 1 in the attack phase.

0 to 8 seconds

**Decay time (MinVel) DECAF**

The time needed to go from 1 to the sustain level in the decay phase.

0 to 8 seconds

**Sustain level (MinVel) SUSTAIN**

The output level during the sustain phase.

0 to 100%

**Release time (MinVel) RELEASE**

The time needed to go from the sustain level to 0 in the release phase.

0 to 8 seconds

**Attack level (MaxVel) ATKLVL^**

The output level at the end of the attack phase.

0 to 100%

**Attack level (MinVel) ATKLVL\_**

The output level at the end of the attack phase.

0 to 100%

**Shape SHAPE**

The shape of the envelope (Log -> Lin -> Exp).

**Velocity follow V.FOLLOW**

Follow the controller's velocity, or use a fixed velocity.

— *Controller*

Follow the velocity of the controller.

— *Min. velocity*

Use a fixed velocity of 1.



- *Max. velocity*

Use a fixed velocity of 127.

Mix (Main bus) (**MAIN**)

**The mixer for the Main bus.**

The Main bus takes four inputs: the three oscillators (**OSC1** to **OSC3**), and another audio input, which can be chosed between white noise and a combination of input ports of the *Anyma V*.

The output of the bus is then sent to the reverb, and it is also sent to the Left and Right channels of the audio output depending on the *Pan* parameter.

**OSC1 Level** **OSC1**

The level of **OSC1** sent to the Main bus.

0 to 100%

**OSC2 Level** **OSC2**

The level of **OSC2** sent to the Main bus.

0 to 100%

**OSC3 Level** **OSC3**

The level of **OSC3** sent to the Main bus.

0 to 100%

**Audio input Level** **AUDIO . IN**

The level of the audio input sent to the Main bus.

0 to 100%

**Level** **LEVEL**

The output level of the Main bus.

0 to 100%

**Reverb Send** REVERB

The level of the Main bus sent to the reverb.

0 to 100%

**Pan** PAN

The balance of the Main bus output to the left and right channels.

100% Left to 100% Right

**Audio input Source** AIN.SRC

An external source for the audio input.

- *White noise*  
A random audio source.
- *Left channel*  
The left channel of the audio input.
- *Right channel*  
The right channel of the audio input.
- *Audio input (sum)*  
Both channels of the audio input summed.

Mix (Aux bus) (**AUX**)

**The mixer for the Aux bus.**

The Aux bus takes four inputs: the three oscillators (**OSC1** to **OSC3**), and another audio input, which can be chosed between white noise and a combination of input ports of the *Anyma V*.

The output of the bus is then sent to the reverb, and it is also sent to the Left and Right channels of the audio output depending on the *Pan* parameter.

**OSC1 Level** **OSC1**

The level of **OSC1** sent to the Aux bus.

0 to 100%

**OSC2 Level** **OSC2**

The level of **OSC2** sent to the Aux bus.

0 to 100%

**OSC3 Level** **OSC3**

The level of **OSC3** sent to the Aux bus.

0 to 100%

**Audio input Level** **AUDIO . IN**

The level of the audio input sent to the Aux bus.

0 to 100%

**Level** **LEVEL**

The output level of the Aux bus.

0 to 100%

**Reverb Send** REVERB

The level of the Aux bus sent to the reverb.

0 to 100%

**Pan** PAN

The balance of the Aux bus output to the left and right channels.

100% Left to 100% Right

**Audio input Source** AIN.SRC

An external source for the audio input.

— *White noise*

A random audio source.

— *Left channel*

The left channel of the audio input.

— *Right channel*

The right channel of the audio input.

— *Audio input (sum)*

Both channels of the audio input summed.

## Reverb (VERB)

### The reverb effect at the end of the signal chain.

The final reverb is a mono effect. It takes its input from the Main and Aux buses, and outputs to both the left and right channels of the audio output.

#### **Bypass** BYPASS

Whether to bypass the reverb, freeing computing resources.

#### **Send** SEND

The level of input.

0 to 100%

#### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

#### **Time** TIME

The amount of reverberation time.

0 to 100%

#### **Diffusion** DIFF

The density of the reverberation.

0 to 100%

#### **Damping** DAMPING

The amount of damping.

0 to 100%

## Arpeggiator (ARP)

### Turns polyphonic note input into arpeggios.

The arpeggiator creates arpeggios from the notes played from a polyphonic controller. When enabled, it replaces controller input in the synth, exactly as if using an external MIDI arpeggiator.

The arpeggio starts at the moment the first note is played, and the pattern unfolds synced to the internal or external clock (depending on your settings and overall setup).

When the arpeggiator is enabled using *Latch*, the last chord played on the controller is arpeggiated indefinitely until another chord is played or *Latch* is removed. If you keep at least one note of the chord playing on your controller, you can also change the chord in realtime by playing other notes: playing existing notes will remove them from the chord, and playing new notes will add them, thus allowing you to gradually change the arpeggio in an interactive fashion.

When the arpeggiator rate is set to *Trigger*, then it will wait for a trigger in channel *Arpeggiator step* (*Arp+*) and the *Gate* and *Swing* parameters will have no effect. To do so, you may want to setup a modulator that outputs a trigger into *Arp+* (e.g., with a *Trigger combinator*, set its output to *Arp+*, and set its first input to an existing trigger).

### Mode MODE

The pattern used to arpeggiate the notes played from the controller.

- *Up*

From lowest to highest note.

- *Down*

From highest to lowest note.

- ***Up/Down***  
Performs the Up and Down patterns in sequence.
- ***Down/Up***  
Performs the Down and Up patterns in sequence.
- ***Up and Down***  
Performs the Up and Down patterns in sequence, repeating the lowest/highest notes.
- ***Down and Up***  
Performs the Down and Up patterns in sequence, repeating the lowest/highest notes.
- ***Converge***  
Starts from the extreme notes and alternates towards the middle.
- ***Diverge***  
Starts from the middle notes and alternates towards the extreme notes.
- ***Converge/Diverge***  
Performs the Converge and Diverge patterns in sequence.
- ***Thumb Up***  
Alternates the lowest note with the Up pattern.
- ***Thumb Up/Down***  
Alternates the lowest note with the Up/Down pattern.
- ***Pinky Up***  
Alternates the Up pattern with the highest note.
- ***Pinky Up/Down***  
Alternates the Up/Down pattern with the highest note.
- ***Order***  
Plays the notes in the order they have been entered.
- ***Random***



Plays the notes in random order.

**Rate** RATE

The speed at which to arpeggiate.

**Gate** GATE

The relative amount of time each arpeggiated note is heard.

**Swing** SWING

The amount of asymmetry in the length of two consecutive notes.

**Repeats** REPEAT

The number of times the pattern is repeated, each time shifted by the distance.

1 to 8

**Distance** DISTANCE

The amount of semitones to shift the pattern at each repeat.

-64 to +63 semitones

**Offset** OFFSET

Sets the first note played in the pattern, rotating the arpeggio.

**Arpeggiator enabled** Enabled

Use the arpeggiator output in place of the direct controller input.

## Matrix (MTX)

### Macros presented in the matrix.

#### **Timbre** TIMBRE

0 to 100%

#### **Color** COLOR

0 to 100%

#### **Chaos** CHAOS

0 to 100%

#### **Position** POS

0 to 100%

#### **Geometry** GEOM

0 to 100%

#### **Material** MATER

0 to 100%

#### **Decay** DECAY

0 to 100%

#### **Resonate** RESO

0 to 100%

#### **Drive** DRIVE

0 to 100%

**Body BODY**

0 to 100%

**Space SPACE**

0 to 100%

**Morph MORPH**

A continuous interpolation between Matrix and Matrix Alt.

0 to 100%

**Source 2 SRC2**

0 to 100%

**FX 1 FX1**

0 to 100%

**FX 2 FX2**

0 to 100%

Matrix (alt.) (MTXalt)

**Alternate version of the matrix (for morphing).**

**Timbre** TIMBRE

0 to 100%

**Color** COLOR

0 to 100%

**Chaos** CHAOS

0 to 100%

**Position** POS

0 to 100%

**Geometry** GEOM

0 to 100%

**Material** MATER

0 to 100%

**Decay** DECAY

0 to 100%

**Resonate** RESO

0 to 100%

**Drive** DRIVE

0 to 100%

**Body** BODY

0 to 100%

**Space** SPACE

0 to 100%

**Source 2** SRC2

0 to 100%

**FX 1** FX1

0 to 100%

**FX 2** FX2

0 to 100%

## Oscillators

Module name	Label
Bow Bowing noise generator	Bow
Wind Wind noise generator	Wind
Strike Percussive noise generator	Strike
White noise A simple white noise generator	Noise
Artin NOIS Filtered noise	NOIS
Artin TWNQ Resonant noise	TWNQ
Artin CLKN Random sample generator	CLKN
Artin CLOU Granular cloud generator	CLOU
Artin PRTC Particle system simulator	PRTC
Artin QPSK Telecommunication data generator	QPSK
Artin TOY* Circuit-bent toy	TOY\*
Corroboree Electro-ethnic noise and percussion	Corrob
Modal resonator Vibrating structure simulator	Modal
String resonator Vibrating string simulator	String
Windsyo Complex reed-based physical models	Windsyo
Brass Brass physical model	Brass
Bass Brass A bass-oriented version of the Brass physical model	BsBrss
Flute A flute-like reed physical model	Flute
Pocket Sax A reed physical model inspired by the pocket sax	PktSax
Cylsax A soprano saxophone-like reed physical model	CylSax
Artin PLUK Simple plucked string	PLUK
Artin BOWD Simple bowed string	BOWD
Artin BLOW Simple single-reed wind	BLOW
Artin FLUT Simple flute	FLUT
Drawbar organ Classic drawbar organ sound	ORGAN
Free Reed Free-reed aerophone model	FreeRd
Bagpipe Bagpipe with a chanter and a drone pipe	Bagpipe

Module name	Label
Overdrive guitar Guitar with distortion and feedback	DrvGtr
Artin BELL Additive bell sound synthesizer	BELL
Artin DRUM Additive metal drum synthesizer	DRUM
Artin KICK 808-style kick drum	KICK
Artin CYMB 808-style cymbal	CYMB
Artin SNAR 808-style snare drum	SNAR
Sine wave Pure tone without any harmonics	Sine
Triangle wave Soft tone with some odd harmonics	Tri
Square wave Harsh, rich tone with many odd harmonics	Square
Sawtooth wave Very rich tone with many harmonics	Saw
Virtual analog A virtual analog oscillator with smooth waveform transition	VA
Artin SUB Waveform with sub-oscillator	SUB
Artin CSAW CS-80 style sawtooth with variable notch	CSAW
Artin MORPH Kobol-style variable waveform (triangle, sawtooth, square, pulse)	MORPH
Artin SAWSQR Blends sawtooth and square wave	SAWSQR
Artin FOLD Folded sine/triangle	FOLD
Artin SYNC Dual hard-synced waveforms	SYNC
Artin X3 Triple oscillator	X3
Artin SAWSWARM Seven detuned sawtooths	SAWSWARM
Artin SAWCOMB Sawtooth and tuned comb filter	SAWCOMB
Artin BUZZ One to many sine waves	BUZZ
Artin VOSM Voice simulator	VOSM
Artin VOWL Early speech synthesizer	VOWL
Artin VFOF FoF vowel simulator	VFOF
Artin HARM Additive synthesizer	HARM
Artin FM Two-operator Sine FM synthesizer	FM
Saw FM Two-operator Saw FM synthesiser	SawFM

Module name	Label
Artin ZPDF CZ-style phase-distorted waveform	ZPDF
Artin WTBL Wavetable synthesizer	WTBL
Artin WMAP 2D wavetable synthesizer	WMAP
Artin WLIN Interpolated wavetable synthesizer	WLIN
Artin WTx4 Four-voice wavetable synthesizer	WTx4
Replicant A copy of the previous oscillator	Replica



Bow

**Bowing noise generator.**

Simulates the raw sound of a bow scratching a material.

**Timbre** TIMBRE

0 to 100%

**Bow pressure** PRESSR

0 to 100%

## Wind

### **Wind noise generator.**

Simulates a variety of continuous blowing, breathing, or wind noises.

#### **Timbre** TIMBRE

0 to 100%

#### **Color** COLOR

0 to 100%

#### **Flow** FLOW

0 to 100%

Strike

**Percussive noise generator.**

Simulates a variety of impulses and percussive noises: hammers, mallets, sticks, plectrums, or particles bouncing on a surface (using the Mallet parameter).

**Timbre** TIMBRE

0 to 100%

**Mallet type** Mallet

The kind of percussive noise generated (samples, impulses, bouncing particles).

0 to 100%

**Trigger** TRIGGER

A trigger used to produce the noise.

White noise (**Noise**)

**A simple white noise generator.**

Artin NOIS (**NOIS**)

**Filtered noise.**

White noise filtered through a resonant multi-mode filter.

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

**Filter resonance** **RESO**

0 to 100%

**Filter mode** **FILTER**

Cross-fades between the low-pass and high-pass outputs of the filter.

0 to 100%

Artin TWNQ (TWNQ)

**Resonant noise.**

White noise filtered through two resonant peak filters with variable distance.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Filter resonance** RESO

0 to 100%

**Peak distance** PEAK

0 to 100%

Artin CLKN (CLKN)

**Random sample generator.**

Produces a looped random sample with variable bit depth.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Cycle length** CYCLE

The length of the generated sample.

0 to 100%

**Quantization** BITS

The quantization level.

0 to 100%

**Trigger** TRIGGER

A trigger used to select a new random sample.

## Artin CLOU (CLOU)

### **Granular cloud generator.**

Produces random noise grains at various temporal intervals.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Grain density/overlap** DENSITY

The density and overlap of the grains.

0 to 100%

#### **Grain randomization** RANDOM

The amount of randomization of the grains.

0 to 100%



Artin PRTC (PRTC)

**Particle system simulator.**

Produces random noise droplets at various temporal intervals.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Particle density/overlap** DENSITY

The density and overlap of the particles.

0 to 100%

**Particle randomization** RANDOM

The amount of randomization of the particles.

0 to 100%

## Artin QPSK (QPSK)

### **Telecommunication data generator.**

Produces sound using a digital modulation method used by fax machines and modems with predetermined data.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Baud rate** BAUD

0 to 100%

#### **Data modulation** MOD

0 to 100%

#### **Trigger** TRIGGER

A trigger used to start a new data frame.

Artin TOY\* (TOY\\*)

**Circuit-bent toy.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Clock rate** CLK.RATE

Lo-fi -> Hi-fi.

0 to 100%

**Bit-crushing** BITCRUSH

Simulates glitches and short-circuits.

0 to 100%

## Corroboree (Corrob)

### **Electro-ethnic noise and percussion.**

Based on the Sylphyo *Corroboree* sound, an expression-controlled noise generator with resonant filters that can turn into a percussion instrument or a hardcore kick.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Drive** DRIVE

0 to 100%

#### **Resonance** RESO

0 to 100%

Modal resonator (**Modal**)

**Vibrating structure simulator.**

A *modal resonator* simulates the resonance of a vibrating structure by describing how it absorbs or reinforces certain frequencies (or modes) in the input signal. The reinforcements at certain modes sustain the oscillations already present in the excitation sound and give the resulting sound its timbre.

The oscillator version of this resonator uses a short burst of noise as its excitation signal, simulating an impact on the resonating structure.

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

**Harmonic structure** **STRUCT**

The nature of the structure (plate, bar, string...).

0 to 100%

**Brightness** **BRIGHT**

The type of material (nylon, wood, steel, glass...).

0 to 100%

**Damping** DAMPING

The amount of damping on the material, making the sound decay slower or faster.

0 to 100%

**Position** POSITION

The position of the excitation point on the structure.

0 to 100%

**Voice count** VOICES

The number of voices.

1 to 3

**Quality** QUALITY

Changes sound quality.

- *Lowest*
- *Low*
- *Medium*
- *High*
- *Highest*

**Trigger** TRIGGER

A trigger used to excite the resonator.

## String resonator (**String**)

### **Vibrating string simulator.**

Simulates the propagation and reflection of a wave in a string.

The oscillator version of this resonator uses a short burst of noise as its excitation signal, simulating an impact on the string.

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Structure** STRUCT

Accentuates the non-linear response of the string.

0 to 100%

### **Brightness** BRIGHT

The type of material used to excite the string.

0 to 100%

### **Damping** DAMPING

The amount of damping on the string, making the sound decay slower or faster.

0 to 100%

**Position POSITION**

The position of the excitation point on the string.

0 to 100%

**Voice count VOICES**

The number of voices.

1 to 3

**Trigger TRIGGER**

A trigger used to excite the resonator.



## Windsyo

### **Complex reed-based physical models.**

Offers a few physical models based on acoustic wind instruments. The models have been pre-tuned and designed to offer a complex response similar to existing acoustic instruments.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Loss filter** LSS . FLT

-100 to 100%

#### **Noise** NOISE

0 to 100%

#### **Growl** GROWL

0 to 100%

#### **AM depth** AM . DEPTH

-100 to 100%

#### **Preset** PRESET

- *Reed One*
- *Flute*
- *Duduk*
- *Sylphinet*

Brass

**Brass physical model.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Timbre** TIMBRE

0 to 100%

**Drive** DRIVE

0 to 100%

**Growl** GROWL

0 to 100%

**Reflection** REFL

0 to 100%

## Bass Brass (BsBrss)

**A bass-oriented version of the Brass physical model.**

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Timbre** TIMBRE

0 to 100%

### **Drive** DRIVE

0 to 100%

### **Growl** GROWL

0 to 100%

### **Reflection** REFL

0 to 100%

Flute

**A flute-like reed physical model.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Growl** GROWL

0 to 100%

## Pocket Sax (PktSax)

**A reed physical model inspired by the pocket sax.**

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Timbre** TIMBRE

0 to 100%

### **Noise** NOISE

0 to 100%

### **Growl** GROWL

0 to 100%

### **Color** COLOR

0 to 100%

## Cylsax (CylSax)

A soprano saxophone-like reed physical model.

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Timbre** TIMBRE

0 to 100%

### **Noise** NOISE

0 to 100%

### **Growl** GROWL

0 to 100%

### **Color** COLOR

0 to 100%

## Artin PLUK (PLUK)

### **Simple plucked string.**

Simulates a plucked string using a physical model based on the Karplus-Strong algorithm.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Damping** DAMPING

The amount by which the vibration is damped.

0 to 100%

#### **Pluck position** POSITION

Low -> High.

0 to 100%

#### **Trigger** TRIGGER

A trigger used to pluck the string.



Artin BOWD (BOWD)

**Simple bowed string.**

Simulates a bowed string using a physical model.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Friction** FRICTION

0 to 100%

**Bow position** POSITION

Low -> High.

0 to 100%

## Artin BLOW (BLOW)

### **Simple single-reed wind.**

Simulates a conical single-reed instrument using a physical model.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Air pressure** PRESSR

0 to 100%

#### **Instrument geometry** GEOM

Determines how the instrument's body acts on the resulting timbre.

0 to 100%

## Artin FLUT (FLUT)

### **Simple flute.**

Simulates a wind instrument excited by an air jet, such as a flute, using a physical model.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Air pressure** PRESSR

0 to 100%

#### **Instrument geometry** GEOM

Determines how the instrument's body acts on the resulting timbre.

0 to 100%

## Drawbar organ (ORGAN)

### **Classic drawbar organ sound.**

The classic electro-mechanical tonewheel organ of the 20th century, with many uses in rock, blues, and jazz.

The underlying model replicates the tuning from the original based on the tonewheel architecture, as well as harmonic foldback, key clicks, and percussion sounds.

By default, the organ is set to the selected *Preset*, and sweeping between presets occurs in a continuous fashion.

If needed, the drawbars can be completely customized by increasing the *Preset/custom mix* to the maximum, and by controlling the *Drawbar* parameters.

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Gain** GAIN

0 to 100%

### **Key click level** CLICK

0 to 100%

**Percussion level PERC**

-100 to 100%

**Preset PRESET**

0 to 100%

**Preset/custom mix MIX**

0 to 100%

**Sub-fundamental 16 '**

The drawbar controlling the amplitude of the sub-fundamental harmonic.

**Sub-third 5.1/3 '**

The drawbar controlling the amplitude of the harmonic at a fifth.

**Fundamental 8 '**

The drawbar controlling the amplitude of the fundamental harmonic.

**2nd harmonic 4 '**

The drawbar controlling the amplitude of the second harmonic.

**3rd harmonic 2.2/3 '**

The drawbar controlling the amplitude of the third harmonic.

**4th harmonic 2 '**

The drawbar controlling the amplitude of the fourth harmonic.

**5th harmonic 1.3/5 '**

The drawbar controlling the amplitude of the fifth harmonic.

**6th harmonic** 1.1/3 '

The drawbar controlling the amplitude of the sixth harmonic.

**8th harmonic** 1 '

The drawbar controlling the amplitude of the eighth harmonic.

## Free Reed (**FreeRd**)

### **Free-reed aerophone model.**

A digital model that reproduces the characteristic timbre of a free reed in a melodica, harmonica or accordion.

The model takes its inspiration from a paper by Matthew Caren et al. (2020)<sup>1</sup>.

#### **Pitch source** **PITCH**

The pitch source to follow.

#### **Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

#### **Timbre** **TIMBRE**

0 to 100%

#### **Color** **COLOR**

0 to 100%

#### **Oversampling** **OVERSPL**

Whether to use oversampling to improve the sound quality.

#### **Note trigger** **TRIGGER**

The trigger used to signal note changes.

---

<sup>1</sup>Caren, Michon and Wright (2020): The KeyWI: An Expressive and Accessible Electronic Wind Instrument. In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME'20), Birmingham, UK. [PDF](#)

## Bagpipe

**Bagpipe with a chanter and a drone pipe.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Timbre** TIMBRE

0 to 100%

**Color** COLOR

0 to 100%

**Drone level** DRONE

The pitch of the drone pipe is determined by the last note played on the chanter when drone level is >0%, or any low note below C4.

0 to 100%

**Note trigger** TRIGGER

The trigger used to signal note changes.



## Overdrive guitar (DrvGtr)

### Guitar with distortion and feedback.

Based on the Sylpho *Electrofuse* sound, an overdriven guitar sound with controllable feedback.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Timbre** TIMBRE

0 to 100%

#### **Dampen** DAMPEN

0 to 100%

#### **Feedback** FEEDBK

0 to 100%

#### **Feedback pitch** FBK.PTC

0 to 100%

Artin BELL (BELL)

**Additive bell sound synthesizer.**

Simulates a bell using a bank of decaying sine waves.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Damping** DAMPING

0 to 100%

**Inharmonicity** INHARM.

0 to 100%

**Trigger** TRIGGER

A trigger used to ring the bell.

Artin DRUM (DRUM)

**Additive metal drum synthesizer.**

Simulates a metal drum using a bank of decaying sine waves with noise.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Damping** DAMPING

0 to 100%

**Brightness** BRIGHT

0 to 100%

**Trigger** TRIGGER

A trigger used to strike the drum.

## Artin KICK (KICK)

### 808-style kick drum.

Produces the typical bass drum sounds found on analog drum machines.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Resonant decay** DECAY

0 to 100%

#### **Tone** TONE

0 to 100%

#### **Trigger** TRIGGER

A trigger used to produce a kick sound.

Artin CYMB (CYMB)

**808-style cymbal.**

Produces the typical cymbal sounds found on analog drum machines.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Filter** FILTER

The cutoff of the filters applied to the signal.

0 to 100%

**Tone** TONE

0 to 100%

## Artin SNAR (SNAR)

### 808-style snare drum.

Produces the typical snare drum sounds found on analog drum machines.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Tone** TONE

0 to 100%

#### **Noise mix** SNAPPY

0 to 100%

#### **Trigger** TRIGGER

A trigger used to produce a snare sound.

Sine wave (**Sine**)

**Pure tone without any harmonics.**

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

Triangle wave (**Tri**)

**Soft tone with some odd harmonics.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents



Square wave (**Square**)

**Harsh, rich tone with many odd harmonics.**

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

**Pulse width** **PWM**

The proportion of time the square is “on” during its period.

0 to 100%

Sawtooth wave (**Saw**)

**Very rich tone with many harmonics.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

Virtual analog (VA)

**A virtual analog oscillator with smooth waveform transition.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Waveform** WAVE

The waveform to use (sine, triangle, sawtooth, square, pulse).

## Artin SUB (SUB)

### Waveform with sub-oscillator.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Pulse width** PWM

The proportion of time the square is “on” during its period.

0 to 100%

#### **Sub-oscillator** SUB

-2 octaves -> No sub -> -1 octave.

0 to 100%

#### **Waveform** WAVE

- *Square*
- *Sawtooth*

Artin CSAW (CSAW)

**CS-80 style sawtooth with variable notch.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Notch width** NOTCH

The width of the notch at the raising edge.

0 to 25%

**Discontinuity depth** DEPTH

Delays the rising edge in either direction, adding a discontinuity.

-100 to 100%

## Artin MORPH (MORPH)

**Kobol-style variable waveform** (triangle, sawtooth, square, pulse).

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Waveform** WAVE

Triangle -> Sawtooth -> Square -> Pulse.

0 to 100%

### **Tone** TONE

Applies low-pass filtering and distortion.

0 to 100%

Artin SAWSQR (SAWSQR)

**Blends sawtooth and square wave.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Timbre** TIMBRE

Controls saw dephasing and square PWM.

0 to 100%

**Waveform** WAVE

Sawtooth -> Square.

0 to 100%

## Artin FOLD (FOLD)

**Folded sine/triangle.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Folding strength** FOLD

Amplifies more so that the waveform folds back onto itself.

0 to 100%

**Waveform** WAVE

Sine -> Triangle.

0 to 100%



Artin SYNC (SYNC)

**Dual hard-synced waveforms.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Sync frequency** SYNC

The frequency interval between master and slave waveform.

0 to 100%

**Balance** BALANCE

Master -> Slave.

0 to 100%

**Waveform** WAVE

- *Square*
- *Sawtooth*

**Pulse width** PWM

The proportion of time the square is “on” during its period.

0 to 100%

## Artin X3 (X3)

### **Triple oscillator.**

Three oscillators with independent pitch, as well as detuning (subtracted to PITCH2 and added to PITCH3).

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Pitch 2** PITCH2

The relative pitch of the second oscillator.

-64 to +63 semitones

#### **Pitch 3** PITCH3

The relative pitch of the third oscillator.

-64 to +63 semitones

#### **Waveform** WAVE

- *Square*
- *Sawtooth*
- *Triangle*

- *Sine*
- *Ring-modulated sine*

**Detune** DETUNE

The amount of detuning between each oscillator.

## Artin SAWSWARM (SAWSWARM)

**Seven detuned sawtooths.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Detune amount** DETUNE

The amount of detuning between each sawtooth.

0 to 100%

**High-pass filter** FILTER

The amount of high-pass filtering applied on the resulting sound.

0 to 100%

Artin SAWCOMB (SAWCOMB)

**Sawtooth and tuned comb filter.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Tuning** TUNING

The transposition applied to the tuned delay line.

-64 to +63 semitones

**Delay feedback** FEEDBK

Negative -> Positive.

0 to 100%

Artin BUZZ (BUZZ)

**One to many sine waves.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Waveform** WAVE

Single sine -> Many sines.

0 to 100%

**Detune** DETUNE

The amount of space between each sine wave.

-100 to +100 cents

Artin VOSM (VOSM)

**Voice simulator.**

Produces sounds inspired by early voice simulators.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Second formant** FMT2

The frequency of the second formant.

-100 to 100%

**Third formant** FMT3

The frequency of the third formant.

-100 to 100%

**Carrier shape** CARRIER

The shape of the carrier.

0 to 100%

## Artin VOWL (VOWL)

### Early speech synthesizer.

Produces lo-fi vowel sounds using simple filtering structures approximating the human vocal tract.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Vowel** VOWEL

A -> E -> I -> O -> U.

0 to 100%

#### **Age** AGE

Old -> Young.

0 to 100%

#### **Consonant** CONSON.

The shape of the consonant to be produced when triggered.

0 to 100%

#### **Consonant length** CONS.LEN

The duration of a consonant when triggered.

0 to 100%



**Consonant trigger** CNS. TRIG

A trigger used to produce a consonant.

## Artin VFOF (VFOF)

### **FoF vowel simulator.**

Produces vowel sounds by approximating the sound of vocal cords and the filtering structure provided by the human vocal tract.

#### **Pitch source** PITCH

The pitch source to follow.

#### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

#### **Vowel** VOWEL

A -> E -> I -> O -> U.

0 to 100%

#### **Age** AGE

Old -> Young.

0 to 100%

Artin HARM (HARM)

**Additive synthesizer.**

A bank of 12 sine waves, each tuned at a multiple of the fundamental frequency, whose amplitude relationships determines the final timbre.

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Central frequency** FREQ

The frequency of the central harmonic.

0 to 100%

**Bandwidth** BW

The distribution of harmonics and presence of the fundamental.

-100 to 100%

## Artin FM (FM)

### Two-operator Sine FM synthesizer.

#### **Pitch source** `PITCH`

The pitch source to follow.

#### **Pitch (semitones)** `SEMI`

The pitch correction on the oscillator.

-64 to +63 semitones

#### **Pitch (fine)** `FINE`

The pitch correction on the oscillator.

-100 to +100 cents

#### **Modulation index** `MOD.IDX`

The amount of modulation of the carrier phase.

0 to 100%

#### **Harmonicity ratio** `RATIO`

The relative frequency interval between modulator and carrier.

0 to 100%

#### **Operator topology** `TPLG`

The arrangement of the feedback paths between modulator and carrier.

- *Classic*

A classic two-operator FM algorithm.

- *Feedback*

Feedback from the carrier to itself.

- *Chaotic feedback*

Feedback from the carrier to itself and to the modulator.

**Harmonicity ratio (fine)** RT.FINE

The relative frequency interval between modulator and carrier (x0.01).

-100 to 100%

## Saw FM (SawFM)

**Two-operator Saw FM synthesiser.**

### **Pitch source** PITCH

The pitch source to follow.

### **Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

### **Modulation** MOD

The amount of modulation.

0 to 100%

### **Feedback** FEEDBK

The amount of feedback.

0 to 100%

### **Crossfade** XFADE

The amount of crossfade (left: saw alone, right: two-operator saw FM).

0 to 100%

Artin ZPDF (ZPDF)

**CZ-style phase-distorted waveform.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Filter cutoff** CUTOFF

The frequency cutoff of the filter.

0 to 100%

**Waveform** WAVE

Sawtooth -> Square -> Triangle.

0 to 100%

**Filter mode** MODE

The type of filter simulated using phase distortion.

- *Low-pass*
- *Peaking*
- *Band-pass*
- *High-pass*

Artin WTBL (WTBL)

**Wavetable synthesizer.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Sweep** SWEEP

Travels through the selected wavetable.

0 to 100%

**Wavetable** WAVETBL

Selects the current wavetable.



Artin WMAP (**WMAP**)

**2D wavetable synthesizer.**

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

**X scanning** **X**

The horizontal position in the grid.

0 to 100%

**Y scanning** **Y**

The vertical position in the grid.

0 to 100%

Artin WLIN (**WLIN**)

**Interpolated wavetable synthesizer.**

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

**Wavetable** **WAVETBL**

Selects the current wavetable.

0 to 100%

**Interpolation** **LERP**

The interpolation method used for playback.

0 to 100%

Artin WTx4 (WTx4)

**Four-voice wavetable synthesizer.**

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the oscillator.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the oscillator.

-100 to +100 cents

**Morph** MORPH

Morphs through wavetables.

0 to 100%

**Detune/chord** CHORD

Detunes the oscillators or selects a fixed chord.

0 to 100%

## Replicant (**Repl**ica)

**A copy of the previous oscillator.**

A replicant is a copy of the previous oscillator, with different expression and pitch settings. This can be used to facilitate the creation of paraphonic patches.

### **Pitch source** **PITCH**

The pitch source to follow.

### **Pitch (semitones)** **SEMI**

The pitch correction on the oscillator.

-64 to +63 semitones

### **Pitch (fine)** **FINE**

The pitch correction on the oscillator.

-100 to +100 cents

## Effects

Module name	Label
Modal resonator Vibrating structure simulator	Modal
String resonator Vibrating string	String
Snare drum A physical model of a snare drum	Snare
Dirty formant filter Old-school FoF-based formant filter	FOFlt
Comb filter Old-school comb filter	CmbFlt
Filter bank A bank of four band-pass filters (BP2 SVF)	FltBnk
State-variable filter Two-stage resonant filter	SVF
Ladder filter Four-stage resonant filter	Ladder
Simple EQ Simple equalizer	EQ
VCA Voltage-controlled amplifier (decreases the level of its input signal)	VCA
Tremolo Change the amplitude of the input in a periodic way	TremL
Noise gate Attenuates the input when the signal is below a threshold	Gate
Compressor Compress the input signal	Comp
Dynamics booster Tame or boost the signal by compressing it	Boost
Cross-fader Balances between two inputs	XFader
Cross-fader with drive Cross-fader with drive controls	XFadeD
Rotary speaker (stereo) Simulation of a stereo rotary speaker (affects both buses)	RotryS
Amplifier A saturating amplifier	AMP
Overdrive Saturates without increasing volume	Drive
Bitcrusher Reduces the resolution of the audio signal	Crush
Cross-folder Wavefolds two inputs together	XFold
Ring modulator Ring-modulates two inputs together	RingMod
XOR modulator XORs two inputs together bit by bit	XOR
CMP modulator Cross-modulates two inputs with digital comparison operators	CMP
Chorus Thickens the input	Chorus

Module name	Label
<b>Phaser</b> Six-stage phase shifter	Phaser
<b>Pitch-shifter</b> Transposes the input	PShifter
<b>Rotary speaker</b> Simulation of a rotary speaker	Rotary
<b>FM Operator</b> An oscillator that can be used to build complex FM structures	FM Op
<b>Delay</b> Delay line with feedback and damping	Delay
<b>Delay (sync)</b> Delay line with feedback and damping (synced to the tempo)	Delay.S
<b>Ping-pong delay</b> Ping-pong stereo delay line (uses both buses)	PngDly
<b>Ping-pong delay (sync)</b> Ping-pong stereo delay line (uses both buses, synced to the tempo)	PingDLS
<b>Reverb</b> Mono reverberation effect	Reverb
<b>Granular processor</b> Creates audio textures by combining short segments of the input	Grains

Modal resonator (**Modal**)

**Vibrating structure simulator.**

A *modal resonator* simulates the resonance of a vibrating structure by describing how it absorbs or reinforces certain frequencies (or modes) in the input signal. The reinforcements at certain modes sustain the oscillations already present in the excitation sound and give the resulting sound its timbre.

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Send** **SEND**

The level of input sent into the effect.

0 to 100%

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Harmonic structure** **STRUCT**

The nature of the structure (plate, bar, string).

0 to 100%

**Brightness** **BRIGHT**

The type of material (nylon, wood, steel, glass).

0 to 100%

**Damping** **DAMPING**

The amount of damping on the material, making the sound decay slower or faster.

0 to 100%

**Position POSITION**

The position of the excitation point on the structure.

0 to 100%

**Voice count VOICES**

The number of voices.

1 to 3

**Pitch source PITCH**

The pitch source to follow.

**Pitch (semitones) SEMI**

The pitch correction on the effect.

-64 to +63 semitones

**Pitch (fine) FINE**

The pitch correction on the effect.

-100 to +100 cents

**Quality QUALITY**

Changes sound quality.

- *Lowest*
- *Low*
- *Medium*
- *High*
- *Highest*



## String resonator (**String**)

### **Vibrating string.**

Simulates the propagation and reflection of a wave in a string.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Send** SEND

The level of input sent into the effect.

0 to 100%

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Structure** STRUCT

Accentuates the non-linear response of the string.

0 to 100%

### **Brightness** BRIGHT

The brightness of the input.

0 to 100%

### **Damping** DAMPING

The amount of damping on the string, making the sound decay slower or faster.

0 to 100%

### **Position** POSITION

The position of the excitation point on the strings.

0 to 100%

**Voice count** VOICES

The number of voices.

1 to 3

**Pitch source** PITCH

The pitch source to follow.

**Pitch (semitones)** SEMI

The pitch correction on the effect.

-64 to +63 semitones

**Pitch (fine)** FINE

The pitch correction on the effect.

-100 to +100 cents

Snare drum (**Snare**)

**A physical model of a snare drum.**

Simulates the propagation and reflection of the input waveform exciting a snare drum with a rattle.

The physical model is derived from a paper by Philipp Schmalfuß et al. (2020)<sup>2</sup> from the Center for Haptic Audio Interaction Research (CHAIR).

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Send** **SEND**

The level of input sent into the effect.

0 to 100%

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Head tuning** **TUNING**

The amount of tension on the drum head.

0 to 100%

**Buzz** **BUZZ**

The amount of tension of the rattle on the snare head.

0 to 100%

---

<sup>2</sup>Schmalfuß, Neupert and Kessler (2020): Efficient Snare Drum Model for Acoustic Interfaces with Piezoelectric Sensors. In Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20), Vienna: Vienna University of Music and Performing Arts. [PDF](#)

**Damping** DAMPING

The amount of damping on high (left) or low (right) frequencies.

-100 to 100%

**Position** POS

The position of the hit on the drum head.

0 to 100%

Dirty formant filter (FOF<sub>Lt</sub>)

**Old-school FoF-based formant filter.**

Produces vowel sounds from the audio input by approximating the filtering structure provided by the human vocal tract.

**Bus BUS**

Whether to apply the effect on the Main or Aux bus.

**Send SEND**

The level of input sent into the effect.

0 to 100%

**Dry/wet DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Vowel VOWEL**

A -> E -> I -> O -> U.

0 to 100%

**Age AGE**

Old -> Young.

0 to 100%

## Comb filter (**CmbFlt**)

### **Old-school comb filter.**

This is a base block for the emulation of stringed instrument sounds.

At its core, this module works by introducing a short delay to a signal, and then feeding it back to itself. The resulting interference causes certain frequencies to reinforce and others to cancel out, leading to a comb-like spectrum. When adjusting feedback closer to the maximum, the sound of a stringed instrument can be approximated.

### **Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

### **Send** **SEND**

The level of input sent into the effect.

0 to 100%

### **Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Delay feedback** **FEEDBK**

Negative -> Positive.

0 to 100%

### **Pitch source** **PITCH**

The pitch source to follow.

### **Pitch (semitones)** **SEMI**

The pitch correction on the effect.

-64 to +63 semitones

**Pitch (fine) FINE**

The pitch correction on the effect.

-100 to +100 cents

## Filter bank (FltBnk)

### **A bank of four band-pass filters (BP2 SVF).**

This is a base block for *modal synthesis*, which replicates the natural resonance frequencies of real-world objects using resonant filters in parallel.

This module offers a bank of four band-pass state-variable filters, with individually controllable pitch, resonance, and level.

By adjusting the pitch and resonance of each filter, you can closely mimic the resonances found in physical materials and instruments.

### **Bus BUS**

Whether to apply the effect on the Main or Aux bus.

### **Pitch source PITCH**

The pitch source to follow.

### **Dry/wet DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Filter 1 Pitch (semitones) SEMI1**

The pitch correction of the first filter relative to the pitch source.

-64 to +63 semitones

### **Filter 2 Pitch (semitones) SEMI2**

The pitch correction of the second filter relative to the pitch source.

-64 to +63 semitones

### **Filter 3 Pitch (semitones) SEMI3**

The pitch correction of the third filter relative to the pitch source.

-64 to +63 semitones



**Filter 4 Pitch (semitones) SEMI4**

The pitch correction of the fourth filter relative to the pitch source.

-64 to +63 semitones

**Filter 1 Pitch (fine) FINE1**

The pitch correction of the first filter relative to the pitch source.

-64 to +63 semitones

**Filter 2 Pitch (fine) FINE2**

The pitch correction of the second filter relative to the pitch source.

-64 to +63 semitones

**Filter 3 Pitch (fine) FINE3**

The pitch correction of the third filter relative to the pitch source.

-64 to +63 semitones

**Filter 4 Pitch (fine) FINE4**

The pitch correction of the fourth filter relative to the pitch source.

-64 to +63 semitones

**Filter 1 Resonance RES01**

The resonance of the first filter.

0 to 100%

**Filter 2 Resonance RES02**

The resonance of the second filter.

0 to 100%

**Filter 3 Resonance RES03**

The resonance of the third filter.

0 to 100%

**Filter 4 Resonance RES04**

The resonance of the fourth filter.

0 to 100%

**Filter 1 Gain LEVEL1**

The output gain of the first filter.

0 to 400%

**Filter 2 Gain LEVEL2**

The output gain of the second filter.

0 to 400%

**Filter 3 Gain LEVEL3**

The output gain of the third filter.

0 to 400%

**Filter 4 Gain LEVEL4**

The output gain of the fourth filter.

0 to 400%

State-variable filter (SVF)

**Two-stage resonant filter.**

Filters the input using a classic two-stage filter capable of producing various responses.

**Bus** BUS

Whether to apply the effect on the Main or Aux bus.

**Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Filter mode** MODE

The type of filter response.

- *Low-pass*  
A low-pass filter (-12 dB/oct).
- *High-pass*  
A high-pass filter (-12 dB/oct).
- *Band-pass*  
A band-pass filter (-12 dB/oct).
- *Notch*  
A notch filter.

**Cutoff frequency** CUTOFF

The cutoff frequency of the filter.

12.26 Hz to 18.82 kHz

**Resonance** RESO

The resonance of the filter.

0 to 100%

**Keyfollow K.FOLLOW**

The amount by which the filter cutoff follows the controller's pitch.

-100 to 100%

Ladder filter (**Ladder**)

**Four-stage resonant filter.**

Filters the input using a classic four-stage filter with variable resonance.

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Filter mode** **MODE**

The type of filter response.

- *Low-pass LP1*  
A low-pass filter (-6 dB/oct).
- *Low-pass LP2*  
A low-pass filter (-12 dB/oct).
- *Low-pass LP3*  
A low-pass filter (-18 dB/oct).
- *Low-pass LP4*  
A low-pass filter (-24 dB/oct).
- *High-pass HP1*  
A high-pass filter (-6 dB/oct).
- *High-pass HP2*  
A high-pass filter (-12 dB/oct).
- *High-pass HP3*  
A high-pass filter (-18 dB/oct).
- *High-pass HP4*  
A high-pass filter (-24 dB/oct).

— *Band-pass BP1*

A band-pass filter (-6 dB/oct).

— *Band-pass BP2*

A band-pass filter (-12 dB/oct).

— *Notch*

A notch filter.

**Cutoff frequency** CUTOFF

The cutoff frequency of the filter.

12.26 Hz to 18.82 kHz

**Resonance** RESO

The resonance of the filter.

0 to 100%

**Keyfollow** K.FOLLOW

The amount by which the filter cutoff follows the controller's pitch.

-100 to 100%

## Simple EQ (EQ)

### **Simple equalizer.**

Adjusts the spectrum of the input with simple equalization controls.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Send** SEND

The level of input sent into the effect.

0 to 100%

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Low-pass** LOWPASS

The low-pass filter frequency.

0 to 100%

### **High-pass** HIPASS

The high-pass filter frequency.

0 to 100%

### **Peak frequency** PEAKFRQ

The peaking filter frequency.

0 to 100%

### **Peak Q** PEAK.Q

The peaking filter Q factor.

0 to 100%

**Peak gain** `PEAKGAIN`

The peaking filter gain.

0 to 100%



VCA

**Voltage-controlled amplifier** (decreases the level of its input signal).

**Bus** BUS

Whether to apply the effect on the Main or Aux bus.

**Level** LEVEL

The level of the input signal.

0 to 100%

## Tremolo (**Trem**l)

**Change the amplitude of the input in a periodic way.**

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Rotation speed** SPEED

The speed at which the amplitude changes.

0 to 100%

### **Amount** AMOUNT

The amount of amplitude modulation.

0 to 100%

Noise gate (**Gate**)

**Attenuates the input when the signal is below a threshold.**

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Sidechain** **SIDECH**

Whether to use the other bus as a sidechain input.

**Threshold level** **THRESH**

The level over which the gate opens.

**Attack time** **ATTACK**

The time to reach the input level when the gate opens.

**Hold time** **HOLD**

The minimum time during which the gate stays open.

**Release time** **RELEASE**

The time to reach the attenuation level when the gate closes.

**Hysteresis amount** **HYST**

The difference between the threshold and the level under which the gate closes.

**Attenuation level** **ATTN**

The level of attenuation applied when the gate is closed.

**Invert** `INVERT`

Whether to invert the gate.

**Lookahead time** `LOOKAHD`

The time by which the input is delayed so the gate can act ahead of transients.

- *None*
- *1.5ms*
- *10ms*

Compressor (**Comp**)

**Compress the input signal.**

A feed-forward compressor that changes the dynamics of the input based on a threshold level.

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Sidechain** **SIDECH**

Whether to use the other bus as a sidechain input.

**Threshold level** **THRESH**

The level above which compression will occur.

**Compression ratio** **RATIO**

The amount of compression.

**Attack time** **ATTACK**

The time for compression to start reacting when above the threshold.

**Release time** **RELEASE**

The time for compression to stop reacting when below the threshold.

**Knee width** **KNEE**

Makes compression happen more gradually around the threshold point.

**Makeup gain** **MAKEUP**

Recovers the amount of level lost by compression.

**Lookahead time** LOOKAHD

The time by which the input is delayed so compression can act ahead of transients.

- *None*
- *1ms*
- *10ms*

## Dynamics booster (**Boost**)

**Tame or boost the signal by compressing it.**

Changes the dynamics of the input.

With a positive amount, the input is boosted; i.e., the signal above a threshold level is compressed, with loudness preserved. With a negative amount, the input is tamed; i.e., the signal above a threshold level is just reduced.

Behind the booster is a compressor with automatic make-up gain (with positive amounts) and automatic attack/release time computation based on transient analysis.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Sensitivity** SENS

Whether the booster reacts to soft signals.

0 to 100%

### **Boost amount** AMOUNT

Tames the signal to the left, boosts it to the right.

## Cross-fader (XFader)

**Balances between two inputs.**

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Mix** MIX

The cross-fading amount (this bus on the left, the other bus on the right).

0 to 100%



Cross-fader with drive (XFadeD)

### **Cross-fader with drive controls.**

#### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

#### **Gain** GAIN

The gain applied to the input signal on this bus.

0 to 400%

#### **Gain 2** GAIN2

The gain applied to the input signal on the other bus.

0 to 400%

#### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

#### **Mix** MIX

The amount of cross-fading after applying drive (this bus on the left, the other bus on the right).

0 to 100%

#### **Oversampling** OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x

Rotary speaker (stereo) (**RotryS**)

**Simulation of a stereo rotary speaker (affects both buses).**

**Bus for input** **BUS**

**Rotation speed** **SPEED**

The speed of the rotating speakers.

0 to 100%

**Amount** **AMOUNT**

The intensity of the effect.

0 to 100%

## Amplifier (AMP)

### **A saturating amplifier.**

Makes the signal louder, simulating the non-linearities found in analog amplifiers.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Gain** GAIN

The gain applied to the input signal.

0 to 400%

### **Oversampling** OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x

## Overdrive (**Drive**)

**Saturates without increasing volume.**

Makes the signal louder, simulating the non-linearities found in analog amplifiers, in a way similar to overdrive guitar pedals.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Drive** DRIVE

The level of overdrive applied to the input signal.

0 to 100%

### **Oversampling** OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x

Bitcrusher (**Crush**)

**Reduces the resolution of the audio signal.**

**Bus** BUS

Whether to apply the effect on the Main or Aux bus.

**Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Depth** DEPTH

The resolution of the output signal.

**Sample rate reduction** REDUX

The downsampling factor of the output signal.

## Cross-folder (XFo~~l~~d)

**Wavefolds two inputs together.**

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Gain** GAIN

The gain applied to the input signal on this bus.

0 to 400%

### **Gain 2** GAIN2

The gain applied to the input signal on the other bus.

0 to 400%

### **Wavefold amount** FOLD

The amount of wavefolding of both signals.

0 to 100%

### **Oversampling** OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x

Ring modulator (**RingMod**)

**Ring-modulates two inputs together.**

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Gain** **GAIN**

The gain applied to the input signal on this bus.

0 to 400%

**Gain 2** **GAIN2**

The gain applied to the input signal on the other bus.

0 to 400%

**Gain boost** **BOOST**

The amount of gain applied to the resulting signal.

0 to 100%

**Analog** **ANALOG**

Whether ring-modulation is analog or digital.

**Oversampling** **OVERSPL**

The oversampling factor.

- 1x
- 2x
- 4x

## XOR modulator (XOR)

**XORs two inputs together bit by bit.**

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Gain** GAIN

The gain applied to the input signal on this bus.

0 to 400%

### **Gain 2** GAIN2

The gain applied to the input signal on the other bus.

0 to 400%

### **Bits to XOR** BITS

Selects which bits are XOR'ed together.

0 to 100%

### **Oversampling** OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x



## CMP modulator (CMP)

**Cross-modulates two inputs with digital comparison operators.**

Creates a new signal by applying a comparison operator to each sample of both inputs ( $s_1$  and  $s_2$ ). It is possible to cross-fade between the following four algorithms:

- If  $s_2 > s_1$ , Then  $s_2$ , Else  $s_1$
- If  $|s_2| > |s_1|$ , Then  $|s_2|$ , Else  $|s_1|$
- If  $|s_2| > |s_1|$ , Then  $|s_2|$ , Else  $-|s_1|$
- If  $|s_1|$ , Then  $|s_1|$ , Else  $|s_2|$

### Bus BUS

Whether to apply the effect on the Main or Aux bus.

### Dry/wet DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### Gain GAIN

The gain applied to the input signal on this bus.

0 to 400%

### Gain 2 GAIN2

The gain applied to the input signal on the other bus.

0 to 400%

### Operator OP

Selects the comparison operator to use.

0 to 100%

### Oversampling OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x

## Chorus

### **Thickens the input.**

Mixes the input with slightly delayed versions of itself.

#### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

#### **Send** SEND

The level of input sent into the effect.

0 to 100%

#### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

#### **Delay time** TIME

The time between the original and the delayed signal.

0 to 26.5 ms

#### **Feedback** FEEDBK

The amount of feedback from the delayed signal.

0 to 100%

#### **Modulation rate** MOD.RATE

The speed of the oscillator modulating delay time.

0.07 to 14.80 Hz

#### **Modulation amount** MOD.AMT

The intensity of the modulation of the delay time.

0 to 100%

## Phaser

### **Six-stage phase shifter.**

Applies a series of six filters that dephase the input, creating continuous motion in its spectrum.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Send** SEND

The level of input sent into the effect.

0 to 100%

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Rate** RATE

The speed of phase shifting.

0.07 to 14.80 Hz

### **Feedback** FEEDBK

The amount of feedback from the phase-shifted signals.

0 to 100%

Pitch-shifter (**PShiftr**)

**Transposes the input.**

Raises or lowers the pitch of the input over a given time window.

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Send** **SEND**

The level of input sent into the effect.

0 to 100%

**Dry/wet** **DRY/WET**

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Pitch** **PITCH**

The relative pitch for the shifted signal.

**Window time** **TIME**

The time of the window over which to shift the signal.

20 to 1000 ms

Rotary speaker (**Rotary**)

**Simulation of a rotary speaker.**

**Bus** BUS

Whether to apply the effect on the Main or Aux bus.

**Rotation speed** SPEED

The speed of the rotating speakers.

0 to 100%

**Amount** AMOUNT

The intensity of the effect.

## FM Operator (FM Op)

**An oscillator that can be used to build complex FM structures.**

This module uses the input as an audio-rate phase modulator for its own internal virtual-analog oscillator, making it an autonomous FM operator in the context of a mixed-FM patch.

The FM operator can use a variety of different carrier waveforms, all of which support through-zero linear FM.

Feedback from the output can also be gradually injected into the input using the *Self-modulation* parameter. To limit the effects of aliasing, a *Damping* parameter allows to reduce modulation index on higher pitches, and the *Oversampling* factor can be increased too (at the cost of a higher patch load).

To replicate an existing FM structure into your patch, you can start by placing up oscillators in OSC1/2/3: those “operators” will not be modulated by anything<sup>3</sup>. Then, you can mix the oscillator’s balances between the Main and Aux buses, and add instances of this modules that will act as the intermediate and/or final operators in the FM structure. This allows you up to 8 FM operators, however due to the fixed nature of the audio signal path, not all topologies are possible.

The volume of the internal oscillator is controlled by Patch Expression, but similarly to OSC1/2/3, an *Expression* mapping destination is also available to control its volume independently. By default, the FM operators will do the right thing, however due to the oscillators and previous operators changing volume when playing and stopping notes, there will

---

<sup>3</sup>Any oscillator can be used, but the *Virtual analog* oscillator could be the best starting point, as it provides the classic waveforms used in FM synthesis.

be a slight modulation effect at these moments. The effect is amplified with a larger modulation index and scale. To get rid of this effect, you might need to disable all mappings to Patch Expression, letting all oscillators and intermediate FM operators play at all times, and only map to the final FM operators' *Expression* mapping destination.

**Bus** **BUS**

Whether to apply the effect on the Main or Aux bus.

**Modulation index** **INDEX**

0 to 100%

**Modulation scale** **SCALE**

The scale factor applied to the modulation index.

1 to 100

**Waveform** **WAVE**

The waveform to use (sine, triangle, sawtooth, square, pulse).

**Pitch source** **PITCH**

The pitch source to follow.

**Pitch (semitones)** **SEMI**

The pitch correction on the effect.

-64 to +63 semitones

**Pitch (fine)** **FINE**

The pitch correction on the effect.

-100 to +100 cents



**Self-modulation** SELFMOD

The amount to which the oscillator modulates itself.

0 to 100%

**Damping** DAMP

Limits the modulation index of higher notes.

0 to 100%

**Through-zero** TZ

Whether the operator can support negative frequencies.

**Oversampling** OVERSPL

The oversampling factor.

- 1x
- 2x
- 4x

## Delay

### **Delay line with feedback and damping.**

Creates a delayed version of the input. The resulting output is damped and mixed with the input again to be passed through the delay line, creating a feedback loop that can be used to create echoes and various effects.

#### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

#### **Send** SEND

The level of input sent into the effect.

0 to 100%

#### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

#### **Delay time** TIME

The time between the original and the delayed signal.

0 to 1000 ms

#### **Feedback** FEEDBK

The amount of feedback from the delayed signal.

0 to 100%

#### **Damping** DAMPING

The amount of damping of the feedback signal.

0 to 100%

Delay (sync) (Delay.S)

**Delay line with feedback and damping (syncd to the tempo).**

**Bus** BUS

Whether to apply the effect on the Main or Aux bus.

**Send** SEND

The level of input sent into the effect.

0 to 100%

**Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Delay time** TIME

The time between the original and the delayed signal.

**Feedback** FEEDBK

The amount of feedback from the delayed signal.

0 to 100%

**Damping** DAMPING

The amount of damping of the feedback signal.

0 to 100%

## Ping-pong delay (PngDly)

**Ping-pong stereo delay line (uses both buses).**

**Send** SEND

The level of input sent into the effect.

0 to 100%

**Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Bus balance** BALANCE

**Delay time** TIME

The time between the original and the delayed signal.

0 to 1000 ms

**Feedback** FEEDBK

The amount of feedback from the delayed signal.

0 to 100%

**Damping** DAMPING

The amount of damping of the feedback signal.

0 to 100%

Ping-pong delay (sync) (**PingDLS**)

**Ping-pong stereo delay line** (uses both buses, synced to the tempo).

**Send** SEND

The level of input sent into the effect.

0 to 100%

**Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

**Bus balance** BALANCE

**Delay time** TIME

The time between the original and the delayed signal.

**Feedback** FEEDBK

The amount of feedback from the delayed signal.

0 to 100%

**Damping** DAMPING

The amount of damping of the feedback signal.

0 to 100%

## Reverb

### **Mono reverberation effect.**

Simulates the reflections and absorptions of sound waves in a space, such as a room or a cave, making the input more persistent.

#### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

#### **Send** SEND

The level of input sent into the effect.

0 to 100%

#### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

#### **Time** TIME

The amount of reverberation time.

0 to 100%

#### **Diffusion** DIFF

The density of the reverberation.

0 to 100%

#### **Damping** DAMPING

The amount of damping.

0 to 100%

## Granular processor (**Grains**)

**Creates audio textures by combining short segments of the input.**

Continuously splits the last second of audio input into small grains, each a few milliseconds long, and plays them back at different speeds and possibly with feedback and other effects, to generate rich soundscapes and textures.

### **Bus** BUS

Whether to apply the effect on the Main or Aux bus.

### **Send** SEND

The level of input sent into the effect.

0 to 100%

### **Dry/wet** DRY/WET

The balance between the pre-effect (dry) and post-effect (wet) signals.

### **Position** POS

How far back in time to select grains.

0 to 100%

### **Grain size** SIZE

The size of the grains.

0 to 100%

### **Grain density** DENSITY

The amount of grains to generate.

0 to 100%

### **Randomize grains** RANDOM

Whether to play the grains at constant rate or at random.

**Grain shape** SHAPE

The envelope shape of the grains (hard/smooth square, ramp up/down, triangle).

0 to 100%

**Diffusion** DIFF

The amount of transient smearing in the grains.

0 to 100%

**Reverse grains** REVERSE

Whether to reverse playback of the grains.

**Pitch** PITCH

The amount of transposition to apply to the grains.

**Feedback** FEEDBK

The amount of output fed back into the input.

0 to 100%



## Modulators

Module name	Label
Simple LFO Simple low-frequency oscillator	LFO
Advanced LFO Advanced low-frequency oscillator with shape and fade-in controls	LFO+
Slow LFO Low-frequency oscillator with very long periods of time	SLOWFO
LFO (sync) Low-frequency oscillator synced to the tempo	LFO.S
Envelope Simple envelope generator	ENV
DAHDSR Envelope DAHDSR envelope generator	DAHDSR
Slow Envelope Slow DAHDSR envelope generator	SlOENV
Step sequencer Change a value according to a predetermined pattern over time	STEPSEQ
Euclidean sequencer Generate euclidean rhythms	EUCLID
Hex sequencer Generate rhythms from hexadecimal numbers	HEXSEQ
Envelope follower Transform an audio signal into a smoothed value	FOLLOW
Timbre follower Extract the brightness from an audio signal	TIMBRE
Drum trigger Derive a signal suitable for percussive sound triggering	DRUM
Xform General-purpose value transform	XFORM
Curve Apply a curve to a value	CURVE
Quantize Reduce the resolution of a value	QTZ
Change polarity Make a unipolar value bipolar, and vice-versa	POLAR
Smooth Smooth out the variations of a value	SMOOTH
Accumulate Accumulate a value or variations over time	ACCU
Lookup table Change a value according to a predetermined shape	LUT16
Bouncing ball Simulate the movement of a single bouncing ball	BOUNCE
Ball impact Simulate the impact of several independent bouncing balls	BALL+
Spring-damper system Simulate a spring-damper system attached to the input	SPRING
Logistic map Unfold the logistic equation on each trigger	LGSMAP

Module name	Label
<b>Tent map</b> Unfold the tent map sequence on each trigger	TNTMAP
<b>Circle map</b> Unfold Arnold's circle map sequence on each trigger	CRCMAP
<b>Discrete chaotic map</b> Apply a specific chaotic map equation on each trigger	DC.MAP
<b>Cellular automaton</b> Use specific bits of a running cellular automaton	CELL
<b>Gate combinator</b> Perform successive operations on a series of gates; e.g., (G1 and G2) or G3	GATEC
<b>Trigger combinator</b> Perform successive operations on a series of triggers; e.g., (T1 and T2) or T3	TRIGC
<b>Gate to trigger</b> Converts gate transitions into triggers	GT>TRIG
<b>Gate delay</b> Delay the gate signal by a given time offset	GDELAY
<b>Trigger delay</b> Delay the trigger signal by a given time offset	TDELAY
<b>Impulse</b> Generate an impulse from a value and a trigger	IMPULSE
<b>Count</b> Count occurrences of a trigger	COUNT
<b>Time</b> Measure the time since a trigger	TIME
<b>Delay</b> Delay the signal by a given time offset	DELAY
<b>Latch</b> Capture a value when a trigger occurs	LATCH
<b>Minimum</b> Keep the minimum of a value since a trigger	MIN
<b>Maximum</b> Keep the maximum of a value since a trigger	MAX
<b>Compare</b> Determine when the input value goes above or below a threshold	CMPR
<b>Clamp</b> Limit a value to an interval	CLAMP
<b>Wrap</b> Wrap a value around an interval	WRAP
<b>Fold</b> Fold a value inside an interval	FOLD
<b>Interpolate</b> Cross-fade between two values	LERP
<b>Interpolate (4-point)</b> Interpolate between four points	LERP4
<b>Calculate</b> Perform successive operations on a series of values; e.g., $\min(I1+I2,I3)$	CALC

Simple LFO (LFO)

**Simple low-frequency oscillator.**

Creates a periodic sinusoidal control signal at a given rate.

**Rate** RATE

The number of periods of the waveform per second.

0.07 to 142 Hz

## Advanced LFO (LFO+)

### Advanced low-frequency oscillator with shape and fade-in controls.

Generates a waveform, such as a sine wave, a square, a triangle, or a random wave, with many possible variations. The waveform can progressively appear or disappear, and its phase can be adjusted.

#### **Rate** RATE

The number of periods of the waveform per second.

0.07 to 142 Hz

#### **Trigger** TRIGGER

A trigger used to reset the LFO.

#### **Shape** SHAPE

The shape of the oscillator waveform.

- *Sine*

Sine wave with 50% variation, wavefolded sine with more extreme values.

- *Triangle / Sawtooth*

Triangle with 50% variation, morphing into sawtooth with more extreme values.

- *Square*

Variation adjusts pulse width.

- *Stepped triangle*

Variation adjusts the number of steps (quantization).

- *Noise*

Variation adjusts the amount of interpolation between noise samples.

#### **Shape variation** SHP . VAR

Sweeps through variations of the oscillator waveform.

0 to 100%

**Fade** FADE

The amount of time necessary for the oscillating waveform to fully appear (left) or disappear (right).

0 to 8 seconds; fade-out to the left, fade-in to the right

**Phase on reset** PHASE

The amount of phase of the waveform when a new cycle starts.

0 to 100%

**Trigger output (reset)** TC.OUT

A trigger output for when the waveform starts a new cycle.

## Slow LFO (SLOWFO)

**Low-frequency oscillator with very long periods of time.**

Generates a slowly-evolving waveform, such as a sine wave, a square, a triangle, or a random wave, with many possible variations and adjustable phase.

### **Period** PERIOD

The time to generate a single cycle of the waveform.

1 day to 1 second

### **Trigger** TRIGGER

A trigger used to reset the LFO.

### **Shape** SHAPE

The shape of the oscillator waveform.

- *Sine*

Sine wave with 50% variation, wavefolded sine with more extreme values.

- *Triangle / Sawtooth*

Triangle with 50% variation, morphing into sawtooth with more extreme values.

- *Square*

Variation adjusts pulse width.

- *Stepped triangle*

Variation adjusts the number of steps (quantization).

- *Noise*

Variation adjusts the amount of interpolation between noise samples.

### **Shape variation** SHP.VAR

Sweeps through variations of the oscillator waveform.

0 to 100%

**Phase on reset** PHASE

The amount of phase of the waveform when a new cycle starts.

0 to 100%

**Trigger output (reset)** TC.OUT

A trigger output for when the waveform starts a new cycle.

LFO (sync) (LFO.S)

**Low-frequency oscillator synced to the tempo.**

**Period** PERIOD

The time to generate a single cycle of the waveform.

**Trigger** TRIGGER

A trigger used to reset the LFO.

**Shape** SHAPE

The shape of the oscillator waveform.

— *Sine*

Sine wave with 50% variation, wavefolded sine with more extreme values.

— *Triangle / Sawtooth*

Triangle with 50% variation, morphing into sawtooth with more extreme values.

— *Square*

Variation adjusts pulse width.

— *Stepped triangle*

Variation adjusts the number of steps (quantization).

— *Noise*

Variation adjusts the amount of interpolation between noise samples.

**Shape variation** SHP.VAR

Sweeps through variations of the oscillator waveform.

0 to 100%

**Phase on reset** PHASE

The amount of phase of the waveform when a new cycle starts.

0 to 100%



**Trigger output (reset) TC.OUT**

A trigger output for when the waveform starts a new cycle.

## Envelope (ENV)

### **Simple envelope generator.**

Creates a typical ADSR envelope starting when a new note is played.

#### **Attack time** ATTACK

The time needed to go from 0 to 1 in the attack phase.

0 to 8 seconds

#### **Decay time** DECAY

The time needed to go from 1 to the sustain level in the decay phase.

0 to 8 seconds

#### **Sustain level** SUSTAIN

The output level during the sustain phase.

0 to 100%

#### **Release time** RELEASE

The time needed to go from the sustain level to 0 in the release phase.

0 to 8 seconds

## DAHDSR Envelope (DAHDSR)

### **DAHDSR envelope generator.**

A slightly more complex version of the classic ADSR envelope, with a variable *delay* before the attack phase starts, and a *hold* phase where the output remains at its maximum value for a variable time after the attack phase and before starting the decay phase.

#### **Gate** GATE

The gate used to control the envelope.

#### **Retrigger** RETRIG

A trigger source used to reset the envelope.

#### **Shape** SHAPE

The shape of the envelope (Log -> Lin -> Exp).

#### **Attack time** ATTACK

The time needed to go from 0 to 1 in the attack phase.

0 to 8 seconds

#### **Decay time** DECAY

The time needed to go from 1 to the sustain level in the decay phase.

0 to 8 seconds

#### **Sustain level** SUSTAIN

The output level during the sustain phase.

0 to 100%

#### **Release time** RELEASE

The time needed to go from the sustain level to 0 in the release phase.

0 to 8 seconds

**Delay time** `DELAY`

The time before the envelope starts its attack phase.

0 to 8 seconds

**Hold time** `HOLD`

The time before the envelope starts its decay phase.

0 to 8 seconds

**Hard reset** `HARD.RST`

Whether the envelope is reset to zero when restarted.

**Trigger output (sustain)** `TC.OUT`

A trigger output for when the envelope starts the sustain phase.

## Slow Envelope (SLoENV)

### **Slow DAHDSR envelope generator.**

A slightly more complex and slower version of the classic ADSR envelope, with a variable *delay* before the attack phase starts, and a *hold* phase where the output remains at its maximum value for a variable time after the attack phase and before starting the decay phase.

#### **Gate** GATE

The gate used to control the envelope.

#### **Retrigger** RETRIG

A trigger source used to reset the envelope.

#### **Shape** SHAPE

The shape of the envelope (Log -> Lin -> Exp).

#### **Attack time** ATTACK

The time needed to go from 0 to 1 in the attack phase.

0 to 64 seconds

#### **Decay time** DECAY

The time needed to go from 1 to the sustain level in the decay phase.

0 to 64 seconds

#### **Sustain level** SUSTAIN

The output level during the sustain phase.

0 to 100%

#### **Release time** RELEASE

The time needed to go from the sustain level to 0 in the release phase.

0 to 64 seconds

**Delay time** `DELAY`

The time before the envelope starts its attack phase.

0 to 64 seconds

**Hold time** `HOLD`

The time before the envelope starts its decay phase.

0 to 64 seconds

**Hard reset** `HARD.RST`

Whether the envelope is reset to zero when restarted.

**Trigger output (sustain)** `TC.OUT`

A trigger output for when the envelope starts the sustain phase.

## Step sequencer (STEPSEQ)

**Change a value according to a predetermined pattern over time.**

A 16-step sequencer that advances in the sequence according to the selected trigger.

The step trigger can be anything, but to function as a standard sequencer, you can use one of the triggers derived from the clock (Sync group).

### **Step count** STEPS

The number of steps in the sequence.

1 to 16

### **Step trigger** TRIG

A trigger used to advance by one step.

### **Reset trigger** RST . TRIG

A trigger used to reset the sequence to the beginning.

### **Step 1** 1

0 to 100%

### **Step 2** 2

0 to 100%

### **Step 3** 3

0 to 100%

### **Step 4** 4

0 to 100%

**Step 5 5**

0 to 100%

**Step 6 6**

0 to 100%

**Step 7 7**

0 to 100%

**Step 8 8**

0 to 100%

**Step 9 9**

0 to 100%

**Step 10 10**

0 to 100%

**Step 11 11**

0 to 100%

**Step 12 12**

0 to 100%

**Step 13 13**

0 to 100%

**Step 14 14**

0 to 100%



### **Step 15** 15

0 to 100%

### **Step 16** 16

0 to 100%

## Euclidean sequencer (EUCLID)

### **Generate euclidean rhythms.**

Creates a rhythmic trigger sequence based on Euclidean rhythms, whose beats are as evenly spaced as possible. As the paper describing them explains, many of those rhythms can be found in many kinds of traditional world music.

#### **Beats** BEATS

The number of beats in the sequence.

#### **Steps** STEPS

The number of steps in the sequence.

#### **Trigger** TRIGGER

A trigger used to advance to the next step in the sequence.

#### **Rotation** SHIFT

The number of steps by which to shift the sequence.

#### **Reset trigger** RST.TRIG

A trigger used to reset to the initial step.

#### **Trigger output** TC.OUT

A trigger output for when the current step has a beat.

## Hex sequencer (HEXSEQ)

### **Generate rhythms from hexadecimal numbers.**

Creates a rhythmic trigger sequence based on the binary notation of a 32-bit hexadecimal number. The hexadecimal number is selected using the eight *Part* parameters, each one controlling a single hexadecimal digit from 0x0 to 0xF, starting with the last digit and counting backwards.

The number of steps of the sequence is then based on the minimum number of bits needed to represent the number. The maximum possible sequence thus has 32 steps.

#### **Digit 0** DIGIT0

A digit of the hexadecimal number.

#### **Digit 1** DIGIT1

A digit of the hexadecimal number.

#### **Trigger** TRIGGER

A trigger used to advance to the next step in the sequence.

#### **Digit 2** DIGIT2

A digit of the hexadecimal number.

#### **Digit 3** DIGIT3

A digit of the hexadecimal number.

#### **Reset trigger** RST. TRIG

A trigger used to reset to the initial step.

**Trigger output** `TC.OUT`

A trigger output for when the current step has a beat.

**Digit 4** `DIGIT4`

A digit of the hexadecimal number.

**Digit 5** `DIGIT5`

A digit of the hexadecimal number.

**Digit 6** `DIGIT6`

A digit of the hexadecimal number.

**Digit 7** `DIGIT7`

A digit of the hexadecimal number.

Envelope follower (**FOLLOW**)

**Transform an audio signal into a smoothed value.**

The output will follow the amplitude of the selected audio input.

**Input** **IN**

The audio input source to use.

- *None*
- *Audio input (Left)*
- *Audio input (Right)*
- *Main bus (post-FX)*
- *Aux bus (post-FX)*

**Rise time** **RISE**

0 to 5 seconds

**Fall time** **FALL**

0 to 5 seconds

**Noise floor** **FLOOR**

0 to 100%

**Gain** **GAIN**

0 to 400%

## Timbre follower (TIMBRE)

**Extract the brightness from an audio signal.**

The output will follow the “brightness” of the selected audio input.

### **Input** IN

The audio input source to use.

- *None*
- *Audio input (Left)*
- *Audio input (Right)*
- *Main bus (post-FX)*
- *Aux bus (post-FX)*

### **Rise time** RISE

0 to 5 seconds

### **Fall time** FALL

0 to 5 seconds

### **Noise floor** FLOOR

0 to 100%

### **Gain** GAIN

0 to 400%

### **Sensitivity** SENS

0 to 100%

Drum trigger (DRUM)

**Derive a signal suitable for percussive sound triggering.**

This is another envelope follower more suitable for percussion.

**Input** IN

The audio input source to use.

- *None*
- *Audio input (Left)*
- *Audio input (Right)*
- *Main bus (post-FX)*
- *Aux bus (post-FX)*

**Attack time** ATTACK

**Release time** RELEASE

**Threshold** THRESH

0 to 100%

**Gain** GAIN

0 to 400%

**Trigger output** TC.OUT

The trigger channel to use as an output.

## Xform (XFORM)

### General-purpose value transform.

Transforms an input value by applying a series of operations. First, the value is constrained between the *Minimum* and *Maximum*. Then, a *curve* is applied to distort it. And finally, the value is smoothed using a slew limiter that can independently smooth the value when it increases or decreases.

#### Input **IN**

The value source to use.

#### Minimum **MIN**

The minimum value of the input to consider.

0 to 100%

#### Maximum **MAX**

The maximum value of the input to consider.

0 to 100%

#### Curve **CURVE**

The amount of curvature to apply to the value.

-100 to 100%

#### Curve type **CURVE**

The kind of curve to apply to the value.

- *Log/Exp*

Log -> Linear -> Exp.

- *Sigmoid*

Sigmoid Horiz. -> Linear -> Sigmoid Vert.



**Rise time** RISE

The amount of smoothing when the value increases.

0 to 5 seconds

**Fall time** FALL

The amount of smoothing when the value decreases.

0 to 5 seconds

## Curve (CURVE)

**Apply a curve to a value.**

### **Input** IN

The value to use.

### **Shape** SHAPE

The shape of the curve to apply.

- *Log/Exp*

Log -> Linear -> Exp.

- *Sigmoid*

Sigmoid Horiz. -> Linear -> Sigmoid Vert.

### **Amount** AMOUNT

The amount of curvature to apply.

-100 to 100%

Quantize (QTZ)

**Reduce the resolution of a value.**

**Input** IN

The value to use.

**Steps** STEPS

The number of different possible values of the output.

1 to 127

Change polarity (**POLAR**)

**Make a unipolar value bipolar, and vice-versa.**

**Input** **IN**

The value to use.

**Polarity change** **CHANGE**

- *Unipolar to bipolar*
- *Bipolar to unipolar*

## Smooth (SMOOTH)

**Smooth out the variations of a value.**

### **Input** IN

An input value.

### **Rise time** RISE

The amount of smoothing when the value increases.

0 to 5 seconds

### **Fall time** FALL

The amount of smoothing when the value decreases.

0 to 5 seconds

### **Shape** SHAPE

Whether smoothing is linear or exponential.

- *Exponential smoothing*

Uses a low-pass filter.

- *Linear smoothing*

Uses linear interpolation.

### **Link times** LINK

When enabled, the rise time will be used to specify both rise and fall times.

### **Reset trigger** RST . TRIG

A trigger used to reset smoothing.

## Accumulate (ACCU)

**Accumulate a value or variations over time.**

### **Input** IN

An input value.

### **Gain** GAIN

An attenuation factor for the input.

0 to 25%

### **High-pass filtering** HIPASS

The amount of high-pass filtering on the input.

0 to 100%

### **Overflow behavior** OVRFLOW

The behavior when the accumulator overflows.

- *Saturate*

Keep the value to the maximum until it is reset.

- *Reset*

Reset to zero upon overflow.

- *Wrap*

Wrap around the value range.

- *Fold*

Go back towards zero upon overflow.

### **Reset trigger** RST . TRIG

A trigger used to reset the accumulator.

### **Trigger output (overflow)** TC . OUT

A trigger output for when the accumulator overflows.

## Lookup table (LUT16)

### **Change a value according to a predetermined shape.**

A 16-value table that can describe any shape. The input value is the index at which to lookup a value in the table, with optional interpolation to account for smooth transitions between two values.

This can be used to apply a custom curve to a value, or to shape more precisely how a modulation animates.

#### **Input** IN

The value to use.

#### **Table size** SIZE

The number of values in the table.

1 to 16

#### **Interpolate** LERP?

Whether to interpolate in-between values.

#### **Value 1** 1

0 to 100%

#### **Value 2** 2

0 to 100%

#### **Value 3** 3

0 to 100%

#### **Value 4** 4

0 to 100%

**Value 5 5**

0 to 100%

**Value 6 6**

0 to 100%

**Value 7 7**

0 to 100%

**Value 8 8**

0 to 100%

**Value 9 9**

0 to 100%

**Value 10 10**

0 to 100%

**Value 11 11**

0 to 100%

**Value 12 12**

0 to 100%

**Value 13 13**

0 to 100%

**Value 14 14**

0 to 100%



**Value 15 15**

0 to 100%

**Value 16 16**

0 to 100%

## Bouncing ball (BOUNCE)

**Simulate the movement of a single bouncing ball.**

Uses Newton's laws to simulate a ball thrown into the air and bouncing against the ground. The output is the vertical position of the ball.

### **Simulation speed** SPEED

The speed of the simulation.

0 to 100%

### **Bounciness** BOUNCE

The restitution coefficient, representing how the ball bounces when hitting the ground.

0 to 100%

### **Trigger** TRIGGER

A trigger used to throw the ball.

### **Initial height** HEIGHT

The height of the ball when thrown.

### **Force** FORCE

The force at which the ball is thrown.

0 to 100%

### **Trigger output** TC.OUT

A trigger output for when a ball hits the ground.

## Ball impact (BALL+)

**Simulate the impact of several independent bouncing balls.**

Uses Newton's laws to simulate up to 8 independent balls falling and bouncing against the ground. The output is the velocity of the balls at impact, and is suitable as an expression input for percussive oscillators.

### **Simulation speed** SPEED

The speed of the simulation.

0 to 100%

### **Bounciness** BOUNCE

The restitution coefficient, representing how a ball bounces when hitting the ground.

0 to 100%

### **Trigger** TRIGGER

A trigger used to throw another ball in the air.

### **Initial height** HEIGHT

The height of a ball when thrown.

### **Throw force** FORCE

The force at which a ball is thrown.

0 to 100%

### **Impact fall time** FALL

The amount of damping of the impulse after impact.

0 to 5 seconds

### **Trigger output** TC.OUT

A trigger output for when a ball hits the ground.

## Spring-damper system (SPRING)

**Simulate a spring-damper system attached to the input.**

Uses Newton's laws and Hooke's law to simulate a spring-damper system attached to the input. The output is the position of the suspended mass.

**Input** IN

An input value.

**Damping** DAMP

The amount of resistance of the spring.

0 to 100%

**Stiffness** STIFF

The stiffness of the spring.

0 to 100%

## Logistic map (LGSMAP)

**Unfold the logistic equation on each trigger.**

The sequence for the logistic map is defined as follows:  $x_{n+1} = rx_n(1 - x_n)$ .

**Reset trigger** RST . TRIG

A trigger used to reset to the initial value.

**Parameter (r)** r

The r parameter of the logistic equation.

**Trigger** TRIGGER

A trigger used to advance to the next value in the sequence.

**Center output** CENTER

Whether to center around  $(r-1)/r$ , making the output bipolar.

## Tent map (TNTMAP)

**Unfold the tent map sequence on each trigger.**

The sequence for the tent map is defined as follows:  $x_{n+1} = \mu x_n(1 - x_n)$ .

**Reset trigger** RST . TRIG

A trigger used to reset to the initial value.

**Parameter (mu)** MU

The mu parameter of the tent map equation.

**Trigger** TRIGGER

A trigger used to advance to the next value in the sequence.

**Center output** CENTER

Whether to center around  $(\mu-1)/\mu$ , making the output bipolar.

## Circle map (CRCMAP)

### Unfold Arnold's circle map sequence on each trigger.

The sequence for the standard circle map is defined as follows:  $x_{n+1} = x_n + \omega + K \sin(2\pi x_n)$ .

#### **Reset trigger** RST . TRIG

A trigger used to reset to the initial value.

#### **Coupling strength (K)** K

The K parameter of the circle map equation.

0 to 100%

#### **Trigger** TRIGGER

A trigger used to advance to the next value in the sequence.

#### **Natural frequency (Omega)** OMEGA

The Omega parameter of the circle map equation.

0 to 100%

## Discrete chaotic map (DC.MAP)

### Apply a specific chaotic map equation on each trigger.

The *sequence equation* parameter smoothly changes the equation used to compute the next value of the sequence, between the Duffing map, the Gingerbreadman map, the Bernoulli map, and the classic Hénon map.

### Reset trigger RST.TRIG

A trigger used to reset to the initial value.

### Sequence equation EQN

The equation to use to compute the next value of the sequence.

### Trigger TRIGGER

A trigger used to advance to the next value in the sequence.

### Initial value INIT

The value when the reset trigger occurs.

0 to 100%



## Cellular automaton (CELL)

### Use specific bits of a running cellular automaton.

A 32-bit elementary cellular automaton whose state is advanced on each trigger according to the *Rule* parameter.

The continuous output value of this module represents the state of the entire automaton, but other outputs can be used to extract the state of specific cells.

### Reset trigger **RST**. TRIG

A trigger used to reset to the initial state.

### Rule **RULE**

The Wolfram code for the rule used to advance the cells.

### Trigger **TRIGGER**

A trigger used to advance to the next value in the sequence.

### Initial state **INIT**

The state when the reset trigger occurs.

### Cell A **CELL.A**

A specific cell number.

### Trig. out. (Cell A) **TRIG.A**

The trigger channel to which to output changes to the cell.

### Cell B **CELL.B**

A specific cell number.

**Trig. out. (Cell B)** `TRIG.B`

The trigger channel to which to output changes to the cell.

**Cell C** `CELL.C`

A specific cell number.

**Trig. out. for Cell C** `TRIG.C`

The trigger channel to which to output changes to the cell.

Gate combinator (GATEC)

**Perform successive operations on a series of gates; e.g., (G1 and G2) or G3.**

**Gate output** GT.OUT

The gate channel to use as output and as a base input.

**Operation 1** OP1

The operation to apply to the base input and input 1.

- *Replace (=)*
- OR
- AND
- XOR
- IMPLY
- NOR
- NAND
- XNOR
- NIMPLY

**Input 1** IN1

An input gate.

**Operation 2** OP2

The operation to apply to the previous result and input 2.

- *Replace (=)*
- OR
- AND
- XOR
- IMPLY

- *NOR*
- *NAND*
- *XNOR*
- *NIMPLY*

**Input 2 IN2**

An input gate.

**Operation 3 OP3**

The operation to apply to the previous result and input 3.

- *Replace (=)*
- *OR*
- *AND*
- *XOR*
- *IMPLY*
- *NOR*
- *NAND*
- *XNOR*
- *NIMPLY*

**Input 3 IN3**

An input gate.

Trigger combinator (TRIGC)

**Perform successive operations on a series of triggers; e.g., (T1 and T2) or T3.**

**Trigger output TC.OUT**

The trigger channel to use as output and as a base input.

**Operation 1 OP1**

The operation to apply to the base input and input 1.

- =
- OR
- AND
- XOR
- IMPLY

**Input 1 IN1**

An input trigger.

**Operation 2 OP2**

The operation to apply to the previous result and input 2.

- =
- OR
- AND
- XOR
- IMPLY

**Input 2 IN2**

An input trigger.

**Operation 3** *OP3*

The operation to apply to the previous result and input 3.

- =
- *OR*
- *AND*
- *XOR*
- *IMPLY*

**Input 3** *IN3*

An input trigger.

Gate to trigger (GT>TRIG)

**Converts gate transitions into triggers.**

**Gate input** `INPUT`

The gate source to use as an input.

**Trigger output (rising edge)** `TC.OPEN`

The trigger channel to output to when the gate opens.

**Trigger output (falling edge)** `TC.CLOS`

The trigger channel to output to when the gate closes.

## Gate delay (GDELAY)

**Delay the gate signal by a given time offset.**

**Gate** `GATE`

The gate source to delay.

**Delay time** `TIME`

The time by which the input is delayed.

0 to 1000 ms



Trigger delay (TDELAY)

**Delay the trigger signal by a given time offset.**

**Trigger** TRIG

The trigger source to delay.

**Delay time** TIME

The time by which the input is delayed.

0 to 1000 ms

## Impulse (IMPULSE)

**Generate an impulse from a value and a trigger.**

The output will be at 100% when the trigger occurs, and gradually fall down to zero.

### **Input** IN

An input value.

### **Trigger** TRIGGER

A trigger used to generate the impulse from the current input value.

### **Fall time** FALL

The time for the impulse to fall down to zero.

0 to 5 seconds

Count (COUNT)

**Count occurrences of a trigger.**

**Trigger** TRIGGER

A trigger.

**Steps** STEPS

The number of steps to count.

1 to 127

**Wrap-around** WRAP

Whether to wrap around when the number of steps is reached.

**Reset trigger** RST . TRIG

A trigger used to reset the count to zero.

**Trigger output (end)** TC.OUT

A trigger output for when the number of steps is reached.

## Time (TIME)

### **Measure the time since a trigger.**

The output will be at 0% when the trigger occurs, and gradually increase to reach 100% in a linear fashion.

### **Trigger** TRIGGER

A trigger.

### **Time** TIME

The time to reach 100% of the output value.

0 to 8 seconds

### **Trigger output (end)** TC.OUT

A trigger output for when 100% of the output value is reached.

Delay (DELAY)

**Delay the signal by a given time offset.**

**Input** `IN`

An input value.

**Delay time** `TIME`

The time by which the input is delayed.

0 to 1000 ms

## Latch (LATCH)

**Capture a value when a trigger occurs.**

The output will be equal to the input value at the last moment the trigger occurred.

**Input** IN

An input value.

**Trigger** TRIGGER

A trigger used to capture the input value.

Minimum (**MIN**)

**Keep the minimum of a value since a trigger.**

The output will be equal to the minimum value of the input from the last moment the trigger occurred.

**Input** **IN**

An input value.

**Trigger** **TRIGGER**

A trigger used to reset to the input value.

## Maximum (MAX)

**Keep the maximum of a value since a trigger.**

The output will be equal to the maximum value of the input from the last moment the trigger occurred.

### **Input** IN

An input value.

### **Trigger** TRIGGER

A trigger used to reset to the input value.



Compare (CMPR)

**Determine when the input value goes above or below a threshold.**

**Input** IN

An input value used for comparison.

**Upper threshold** THRES>

The value above which the signal must go.

0 to 100%

**Lower threshold** THRES<

The value below which the signal must go.

0 to 100%

**Gate output** GT.OUT

A gate output for when the signal is above the upper threshold.

**Gate operation** GT.OP

The operation to use to combine the new gate output to the existing value.

- *Replace (=)*
- *OR*
- *AND*
- *XOR*
- *IMPLY*
- *NOR*
- *NAND*
- *XNOR*
- *NIMPLY*

**Trigger output (above) TC.OUT>**

A trigger output for when the signal goes above the upper threshold.

**Trigger output (below) TC.OUT<**

A trigger output for when the signal goes below the lower threshold.

## Clamp (CLAMP)

### Limit a value to an interval.

Clamps the input between the values indicated by *Minimum* and *Maximum*. If the input is less than *Minimum*, the output will be *Minimum*. If the input is greater than *Maximum*, the output will be *Maximum*.

#### **Input** IN

The value to use.

#### **Minimum** MIN

The minimum value.

0 to 100%

#### **Maximum** MAX

The maximum value.

0 to 100%

## Wrap (WRAP)

### **Wrap a value around an interval.**

Wraps the input around the interval indicated by *Minimum* and *Maximum*. For instance, if *Minimum* is 0.5 and *Maximum* is 1, then 0.25 will become 0.75.

#### **Input** IN

The value to use.

#### **Minimum** MIN

The minimum value.

0 to 100%

#### **Maximum** MAX

The maximum value.

0 to 100%

## Fold (FOLD)

### **Fold a value inside an interval.**

Folds the input inside the interval indicated by *Minimum* and *Maximum*. For instance, as the input decreases continuously below *Minimum*, the output will increase in mirror until it reaches *Maximum*, at which point it will decrease, and so on, so as to always stay between *Minimum* and *Maximum*.

#### **Input** IN

The value to use.

#### **Minimum** MIN

The minimum value.

0 to 100%

#### **Maximum** MAX

The maximum value.

0 to 100%

## Interpolate (LERP)

### **Cross-fade between two values.**

Takes two inputs and outputs a point in-between these two values, doing a linear interpolation.

#### **Input 1** IN1

The start value.

#### **Input 2** IN2

The end value.

#### **Interpolation position** LERP

The mix between both values.

Interpolate (4-point) (LERP4)

**Interpolate between four points.**

Interpolates linearly between a series of four points, which can be used to create custom envelopes or waveforms.

**P1 Position P1.P0S**

The position of point P1.

0 to 100%

**P2 Position P2.P0S**

The position of point P2.

0 to 100%

**Interpolation position LERP**

The mix between the values defined at 0%, P1, P2, and 100%.

**Start value 0%**

The value when the interpolation is at 0%.

-100 to 100%

**Value at P1 P1%**

The value when the interpolation is at P1.

-100 to 100%

**Value at P2 P2%**

The value when the interpolation is at P2.

-100 to 100%

**End value 100%**

The value when the interpolation is at 100%.

-100 to 100%

## Calculate (CALC)

**Perform successive operations on a series of values; e.g., min(I1+I2,I3).**

### **Input 1 IN1**

An input value.

### **Operation 1 OP1**

The operation to apply to inputs 1 and 2.

- *Add*

Add both operands.

- *Subtract*

Subtract both operands.

- *Multiply*

Multiply both operands.

- *Minimum*

Take the minimum of both operands.

- *Maximum*

Take the maximum both operands.

### **Input 2 IN2**

An input value.

### **Operation 2 OP2**

The operation to apply to the previous result and input 3.

- *Add*

Add both operands.

- *Subtract*

Subtract both operands.



- *Multiply*

Multiply both operands.

- *Minimum*

Take the minimum of both operands.

- *Maximum*

Take the maximum both operands.

### **Input 3 IN3**

An input value.

### **Operation 3 OP3**

The operation to apply to the previous result and input 4.

- *Add*

Add both operands.

- *Subtract*

Subtract both operands.

- *Multiply*

Multiply both operands.

- *Minimum*

Take the minimum of both operands.

- *Maximum*

Take the maximum both operands.

### **Input 4 IN4**

An input value.

## Mappings

### **Source** FROM

The value that will be used to modulate a parameter.

### **Destination module** TO

The module where the destination parameter is to be found.

### **Destination parameter** TO.PARAM

The parameter to be modulated.

0 to 64

### **Amount** AMOUNT

The amount by which the source modulates the destination parameter.

-100 to 100%

### **Minimum value** MIN

The minimum source value.

0 to 100%

### **Maximum value** MAX

The maximum source value.

0 to 100%

### **Curve amount** CURVE

The amount of curvature to apply to the source value.

-100 to 100%

### **Smoothing amount** SMOOTH

The amount of smoothing to apply to the source value.

0 to 5 seconds

**Scale** `SCALE`

The scale of the mapping amount.

0 to 100%

**Sidechain** `SIDE`

A value that can be used to modulate the amount.

**Sidechain amount** `SIDE.AMT`

How much the sidechain modulates the amount.

-100 to 100%

©2024 Aodyo.  
All rights reserved.

Aodyo SAS  
11B avenue de l'Harmonie  
59650 Villeneuve d'Ascq  
France

[contact@aodyo.com](mailto:contact@aodyo.com)  
[www.aodyo.com](http://www.aodyo.com)

