AODYO INSTRUMENTS

anyma φ

User manual

VERSION 0.8.90
JULY 8, 2021
www.aodyo.com

AODYO INSTRUMENTS

www.aodyo.com

Contents

Overview

A warm thank you from the Aodyo team for believing in us and supporting our work! We hope you will love using your **Anyma Phi** as much as we do.

Anyma Phi is a hybrid monophonic synthesizer that allows you to create and play with new sound universes inspired from the real world, where objects collide, vibrate, and resonate.

Anyma Phi 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.

Anyma Phi is compact enough to stand on your keyboard, or anywhere on your desk, and is compatible with any MIDI controller, such as keyboards, sequencers, or even wind controllers, including our own Sylphyo.

Whether you play live or record in the studio, design sounds or play presets, Anyma Phi will become an indispensable tool. It can even be used as a stand-alone effects processor, or as part of a guitar effects pedal setup.

No matter your level of expertise, synth veteran or total novice, you will find that Anyma Phi 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.

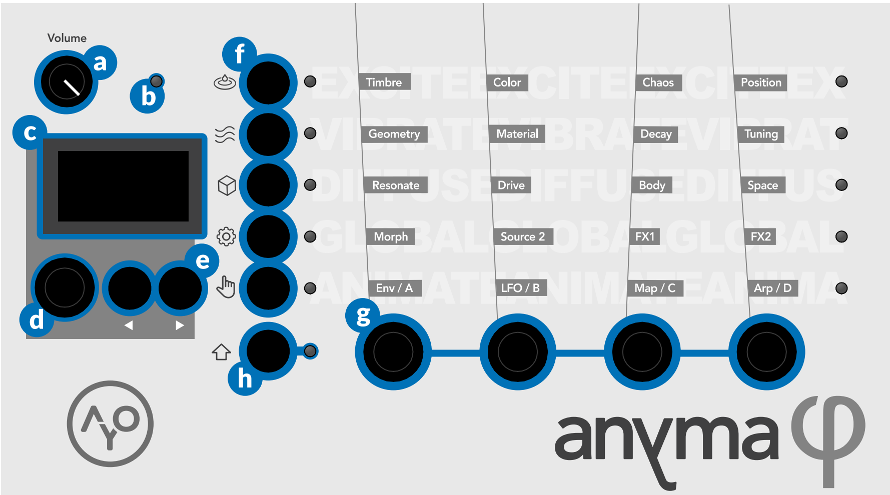
The Anyma Phi is a *monophonic* synthesizer, meaning it is designed to play a single voice at a time; that is, a melody played by a single instrument player or sequencer, generally without chords or harmony[1].

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 your Anyma Phi. 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*.

[1] It is possible to create patches that play chords, or to use an arpeggiator, but remember that the synth is still designed to be controlled a single note at a time.

Front panel



a Volume knob

Controls the output volume of your Anyma Phi.

b Patch LED

Indicates the color of the current patch.

c Display

Shows the currently accessible parameters that can be modified using the encoders, as well as relevant information or menus.

d Display encoder D

While in the matrix, turn the display encoder to select another patch.

In menus and other screens, turn it to select another element.

Press it from anywhere to go to the *settings menu*.

e **Previous and Next buttons** ◀ ▶

Press the ◀ or the ▶ button to switch to the previous or next screen.

Inside a menu, press ▶ to confirm your selection, and ◀ to go back.

Outside of a menu, hold ⇧ and press the ▶ button to save the current patch.

f **Matrix buttons**

While in the matrix, press the buttons to switch to the corresponding line of the matrix: *Excite*, *Vibrate*, *Diffuse*, *Global*, and *Animate*.

In a module editor screen, press the illuminated matrix buttons to access more parameters for the selected module.

g **Parameter encoders** 1 2 3 4

Turn the encoders to modify the parameters described in the bottom part of the display.

Press an encoder to access related mappings or go to its parent module.

h **Shift button** ⇧

While in the matrix, press the ⇧ button to shift to the alternative version of the matrix.

Also provides various shortcuts to edit patch modules.

PERCUSSIVE PLAYING SURFACE

When no external input is plugged into your Anyma Phi, the right audio input is automatically connected to the internal piezoelectric contact microphone that is placed behind the ^Yo logo on the front panel.
When audio input is used in a sound (which is the case, for instance, with patch #7 `Piezo Pling`), this means you can tap and scratch around on the entire case of your Anyma Phi to create a wide variety of percussive effects.

Rear panel

a **Headphone output port** *(6.35mm stereo jack)*
Play the Anyma Phi with headphones.

b **Line input port** *(6.35mm stereo jack)*
Use any audio input as a supplementary sound source for your patches.
When the line input port is plugged in, the internal percussive surface cannot be used.

c **Line output ports** *(left and right 6.35mm mono jacks)*
Play the Anyma Phi on an amp, a mixer, or a hi-fi system.

d **MIDI ports** *(DIN)*
Connect a MIDI keyboard, controller, or device to play or control the Anyma Phi (*MIDI in*), or route MIDI from the USB device port to an external MIDI device (*MIDI out*).

e **USB host port** *(USB A)*
Connect a USB-MIDI keyboard, controller, or device to play or

control the Anyma Phi, or route MIDI from the USB device port to an external MIDI device.

f **USB device port** *(USB Mini-B)*
Connect your Anyma Phi to a computer to update its internal software, to edit sounds with the PC/Mac editor, or to control it from a DAW.

g **Power supply port** *(DC 5V 1A, center positive)*
Provides power to your Anyma Phi, using the included cable and power supply.

h **Power switch**
Toggle the power switch to turn your Anyma Phi on and off.
In order to avoid undesired noises, always make sure you first lower the volume on the mixer, amp, or speakers it is connected to.

Included accessories

a **Power supply cable**
Connect the included power supply to the power supply port of your Anyma Phi.

b **Power supply** *(10.5 W)*
Plug the power supply into a mains socket in order to power your Anyma Phi.

c **USB cable** *(USB Mini-B)*

Use the USB cable to connect your Anyma Phi to your computer in order to update its internal software, to edit sounds with the PC/Mac editor, or to control it from a DAW.

First steps

Connect your Anyma Phi to the mains using the included cable and power supply. The cable must be connected to the USB port of the power supply, and to the power supply port labeled *DC 5V 1A* of the Anyma Phi at the other end.

To play most sounds produced by your Anyma Phi, you will need to use a MIDI keyboard or controller. **Connect** it to the Anyma Phi, either through USB via its *USB Host* port, or by using a MIDI DIN cable plugged into the *MIDI IN* port.

You will also need to **connect** your Anyma Phi to a mixer, an amp, speakers, or headphones, using the relevant audio output ports on the back of your Anyma Phi.

Now, **turn on** your Anyma Phi using the *power switch* on the back. The display will light up, and after a few seconds you will see the main screen.

You can now **play** notes from your MIDI keyboard or controller and hear the sound through your audio system.
If you can't hear anything, **adjust** the volume using the *volume knob*.

Try to **change** the timbre your sound by turning any of the *parameter encoders* (1) (2) (3) (4), and use the *matrix buttons* to reach other controls that impact your sound in sometimes drastic ways. 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*. Your Anyma Phi can contain up to 200 different patches, allowing as many different sonic universes and experiences. It's up to you to decide how you will fill all this available space, be it by downloading patches made by others, or by creating your own.

Initially, your Anyma Phi 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 turning the *display encoder* D, and feel free to play and explore its variations[2].

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

Your Anyma Phi 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, the Anyma Phi 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

[2]Note that all your previous modifications will be lost when you switch to another patch, unless you explicitly save them by pressing ⇧ + ▶.

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.

Updating your Anyma Phi

Chances are that while you are reading this, a new version of the internal software of your Anyma Phi is available. Thanks to updates, your Anyma Phi will keep evolving over the years.

The latest update is always available *on our website*. Regularly check it, or subscribe to our newsletter, to stay informed of new updates.

Please make sure you download the update package corresponding to the operating system version of your computer.

If you're using a computer running another operating system, such as Linux, or a tablet, you will still be able to update your Anyma Phi, but you will not be able to use the editor application. In this case, just download the update package for Mac and perform the update as indicated below.

Each update package contains the following:

- the internal software for your Anyma Phi, packaged as a `.ayo` update file that you will use during the update procedure,
- the user manual for that specific version in the PDF format,

- the editor application you can use after the update to edit and manage your patches from your computer,
- and a `readme.txt` file containing all the instructions for updating and getting started with the editor application.

To update your Anyma Phi, please use the following instructions:

- Power off your Anyma Phi.
- Power on your Anyma Phi while holding the ⇧ button. The display should read `ANYMA PHI UPDATER`.
- Connect the *USB DEVICE* port of your Anyma Phi to your computer using a mini-USB cable.
- Your computer should detect a new removable disk named `ANYMAPHI`.
- Drop the `.ayo` update file from this package to the disk.
- The update will begin. You can see the progress on the display of your Anyma Phi[3].
- If the update has not begun, try ejecting the `ANYMAPHI` disk from your computer using your file explorer. This should start the update.
- Once the update is done, the display will read `UPDATE DONE`.
- You can now eject the `ANYMAPHI` disk from your computer.
- Once the disk has been ejected, you can turn off your Anyma Phi, and turn it on again to enjoy your new update.

If there is any problem during the update, just follow the instructions again from the beginning. If you're still not able to update, please contact us at `support@aodyo.com`.

[3] Do not worry if your file explorer seems to have freezed, it will return to normal as soon as the update is over (about 1 minute).

The matrix

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

Introduction

The starting point of your Anyma Phi is its **matrix**, which provides a quick and easy way to tweak your sounds by following a physical metaphor, with four parameters for each of the five 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*, *Decay*, and *Tuning*
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).

Animate *Envelope*, *LFO*, *Map*, and *Arp*
How the sound moves and comes to life.

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 Phi users, allowing you to more easily tweak any sound you can load into your Anyma Phi 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[1], 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.

The *Animate* line

The last line of the matrix, *Animate*, operates a bit differently from the ones above: instead of providing direct access to controls you can

[1]Which hopefully you'll have a blast doing very soon!

directly manipulate, it allows you to monitor the different *sources* of variations in your patch. These variations can come from internal modulators like envelope generators and low-frequency oscillators (LFO), or from an external source, such as your MIDI keyboard or controller.

Each of the three first parameters of the *Animate* line allows you to focus on a specific source, each with a different list[2]:

(1) *Env* **Envelope generators**
Scroll through all the envelope generators in the patch.
(2) *LFO* **Low-frequency oscillators**
Scroll through all the LFOs in the patch.
(3) *Map* **Mapping sources**
Scroll through all the variation sources (internal or external) in the patch.

Turning one of the parameter encoders allows you to select a particular source, which changes the corresponding name and value visible at the bottom of the screen, and also displays a scope view of the signal generated by the source.

You can also press the parameter encoder to focus on its selected source without scrolling through the list[3].

External controls (A/B/C/D)

If you press *Animate* a second time, you will switch to a view of the four external controls, called *A*, *B*, *C*, and *D*. These controls correspond

[2]At this point, Arp is not used, but is planned for a future update.
[3]If you make a long press by mistake, another screen might appear, but as we won't cover in-depth patch editing just yet, press the ◀ button to return to the matrix.

to actual MIDI messages, especially *control changes* (CC), that your MIDI keyboard or controller can send to your Anyma Phi.

When viewing external controls, you will see the **CTRL** tab highlighted at the top of the display. You will also see a scope view of whichever external control was last selected. To select a different external control, press the corresponding parameter encoder[4]. You can also turn a parameter encoder to directly change the value of A/B/C/D, resulting in exactly the same change in sound as what would have happened if you were to change the value of A/B/C/D using your MIDI controller[5].

To go back to the *Animate* line, just press again, and you will see the **ANIMATE** tab highlighted at the top of the display.

We will see later in this chapter how to setup your Anyma Phi to map A/B/C/D to specific MIDI messages sent by your keyboard or controller. By default, the setup is as follows:

External control	is mapped to...	and...
A	CC 1 Mod. wheel	
B	CC 75	CC 74
C	CC 76	AT Aftertouch[6]
D	CC 77	

[4] Again, don't forget that if you press the encoder for too long and go to a weird screen, you can still press to go back to the matrix.

[5] But keep in mind that moving the value of A/B/C/D directly using a parameter encoder will not bring this change back to your MIDI controller. The change will concern your Anyma Phi only.

[6] While not technically a MIDI CC message, you can map any external control to *channel aftertouch* for convenience.

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*.

In some sounds, *Morph* can even move according to an external control[7], 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 lights on the matrix will then blink until you press ⇧ again to return to the *normal* state. 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.

[7] In the 7 first sounds that came with your **Anyma Phi**, 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 your **Anyma Phi**. If you're already familiar with synthesizers, you might be tempted to skip to the next chapter. However, we recommend that you don't because it also describes the specific way the **Anyma Phi** 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 edit your sounds this way either directly on your Anyma Phi, or using the Mac or PC editor that you can *download on our website*.

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*).

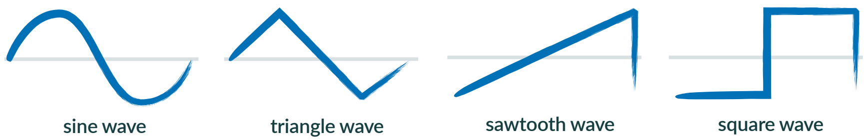
Your Anyma Phi is such a synthesizer, so its responsibility is to understand what a controller sends it, and to turn it into sound, so that the whole formed by the controller-synthesizer couple feels like an expressive and powerful instrument. Now, where does the sound come from?

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 the Anyma Phi, 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).

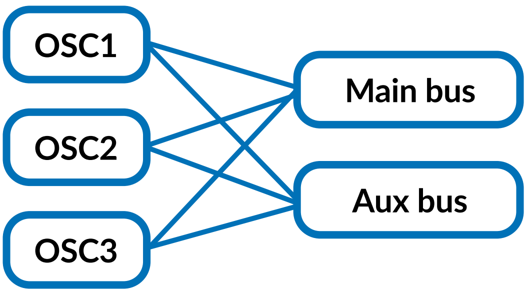
In the Anyma Phi, you can use up to five different effect modules, with an additional final *Reverb* effect that applies to the final sound as a whole. Now, these effects don't do anything in isolation, as they need to take sound from somewhere. So how are oscillators and effects interconnected?

Audio signal path

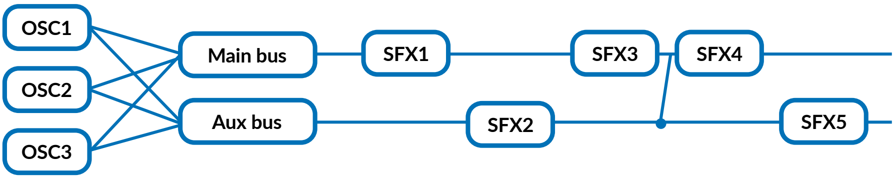
The way sound flows from oscillators to effects to the synth's output will be referred to as the **audio signal path**. There are many different philosophies for determining the signal path: modular synthesizers allow you to plug oscillators and effect modules freely, while more traditional synthesizers 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 predictible places and no time is spent connecting all modules together.

Your Anyma Phi 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.

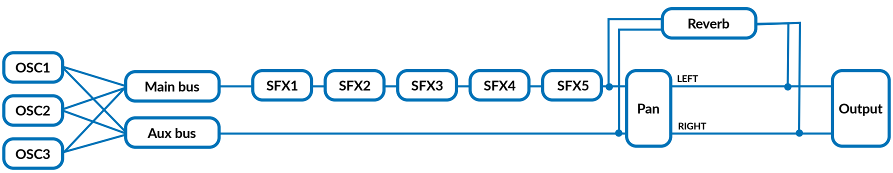
Each of the three oscillators (OSC1, OSC2, OSC3) outputs its sound into two **buses**: the **Main bus** and the **Aux bus**. You can freely choose how much of each oscillator is in either bus.



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 to 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 your Anyma Phi. 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[1].

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

[1] Even another mapping! But let's not get carried away too soon.

the signal over time, or even to make it depend on another *sidechain* source to build complex relationships and interdependencies.

Your Anyma Phi 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.

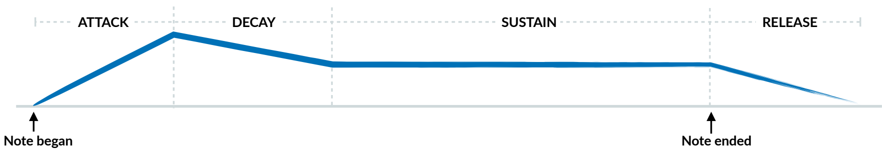
But in musical instruments, not all the work is done by the performer. Sometimes, things oscillate by themselves, act seemingly randomly, or take a bit of time to appear. To be able to reproduce such phenomena, we would need to create new control signals which can be used as sources in mappings, but 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[2]:

- 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.



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.

[2]But keep in mind that there are many variations of this principle out there.

In addition to envelopes and LFOs, the Anyma Phi 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. The Anyma Phi allows up to 16 modulators.

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 *Decay* in the 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 *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 signalsn act upon the patch in real time,
- and the matrix plus its alternate version.

GENERAL							
Patch *Tuning, volume, color...*	VelEnv *Velocity envelope*	Main *Mix for Main bus*	Aux *Mix for Aux bus*	Verb *Final reverb*		Mtx *Matrix controls*	MtxAlt *Alternate matrix controls*
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[3]. It completely describes a sound and how it reacts to your controller and to the matrix.

Some modules, like the Main and Aux buses, cannot be turned off, while the oscillators, effects, modulators, and mappings can. You are free to use as many modules as you like, but when a patch becomes too resource-intensive you might experience audio glitches and dropouts.

[3] Up to 960 in a very stuffed patch!

When this happens, the display of your Anyma Phi will show a little **!!** icon, and you will need to turn off an oscillator or effect to release resources. When using the PC/Mac editor, you can always keep an eye on the amount of resources used by a patch.

The bank

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

You can only store a single bank in your Anyma Phi, but in the PC/Mac editor, 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.

Editing patches

By centering on its matrix, the **Anyma Phi** 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.

Zooming on a parameter

The most important action you can learn for editing patches is **zooming** on a parameter. When you're anywhere in the matrix, or in any edition screen, if you want to see what's behind one of the four parameters described at the bottom of the screen, you can *zoom* on it by holding its corresponding encoder 1 2 3 4 pressed for half a second.

When you *zoom* on something, you access a new screen with more details related to it.

For instance, zooming on *Timbre* displays all the mappings related to it, with each mapping describing the effect *Timbre* has on a single parameter somewhere in the patch.

Let's go to patch #1 `Laraaji +`, select the *Excite* line on the matrix, and **zoom** on *Timbre* by pressing 1 for half a second.

The screen that appears shows us that *Timbre* is mapped to two parameters, because there are two tabs, and that the first mapping describes how by moving *Timbre* to 100% will shift the *Damping* of the first oscillator (*OSC1*) by a small amount to the right.

Here you can already adjust the amount, or even the initial position of *Damping*, by turning the corresponding encoder.

Zooming recursively

Now, let's imagine the effect of *Timbre* isn't quite right and we need to tweak *OSC1* more globally, perhaps by adjusting its other parameters, so that a high value of *Timbre* will sound more harmonious. To do so, **zoom** on *Damping* 4.

You are now inside **OSC1**, with a focus on the *Damping* parameter, and you can then tweak the *Position* parameter to the right of *Damping*.

Zooming allows you to move around the patch by following the relationships between specific elements.

For example, you might wonder what else affects *Position*, which you can see moving from its base value when you play notes on your MIDI keyboard. **Zooming** on *Position* 2 will display the two mappings that affect this parameter.

The first one, **MAP4**, shows that the *expression* or *velocity envelope* will control it over its entire range.

By **turning** the *display encoder* Ⓓ, you can access the second mapping, **MAP9**, which shows that the *Position* control of the matrix can also be used to control the parameter over its entire range.

Navigation

Now, having zoomed three times already, you've shifted your focus away from your initial concern, which was *Timbre*. But you can always recover your train of thought by going back to the previous screens: just press ◀ as many times as needed to return where you were previously, until you finally return to the matrix[1], your starting point. Should you need to return to one of those screens, you can always use ▶.

The ◀ and ▶ buttons operate somewhat similarly to those of a web browser, allowing you to quickly move across a web of related things, with the matrix as your home page.

Mapping creation

Let's return to the mappings related to *Timbre*, for example by pressing ⇧ + ◀, then ▶. It would be nice that when *Timbre* increases,

[1] You can also use ⇧ + ◀ to return directly to the matrix.

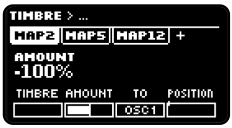
the *Position* parameter of *OSC1* decreases a bit. To do so, we'll create a new mapping from *Timbre*.

Turn the *display encoder* D all the way to the right so as to select the + tab. Then, press the ▶ button.

You are presented with a list of all the modules in the patch, and your task is now to select one. To do so, **turn** the *display encoder* D until **OSC1** appears, and then press ▶.

You now have another list with all the parameters of *OSC1* that can be affected by the mapping. Select **Pluck position** by turning the *display encoder* D, and press ▶ to confirm.

You are now back to the mapping list, and a new one, `MAP2`, has appeared. This is the one we just created. **Turn** 2 to set its *amount* to -100%, so that when *Timbre* increases, the *Position* of *OSC1* will decrease by the same amount.



Then, press ⇧ + ◀ to go back to the matrix, and try your modifications by moving *Timbre*.

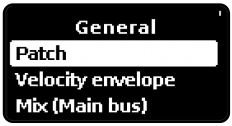
If you're content with them, you can **save** the patch using ⇧ + ▶, so that your modifications will not be forgotten once you turn off your Anyma Phi or switch to another patch.

Jumping to a module

In some cases, you already know exactly which parameter you want to tweak, and you just want to jump to the module it pertains to.

To do so, **hold** the ⇧ button pressed, and start turning the *display encoder* D, without releasing ⇧ yet.

The list of all modules will appear on the display, ordered by category. **Scroll** until you find the module you're looking for, and **release** ⇧ to confirm.



QUICK ACCESS

You will see the lights on the matrix change when scrolling to a different category; e.g., from **General** to **Oscillators**. While still holding ⇧, pressing on of the matrix buttons is another way to quickly go to a specific category.
In this case, pressing ≋ would be a quicker way to access **OSC1**. Subsequent presses, or leaving ≋ pressed, will allow you to cycle through all the oscillators. When your choice is made, you can then release ⇧ to confirm.

The module editor screen

After jumping to a module or, in some cases, zooming on a parameter, you will land on a **module editor screen** focused on a specific module.

The module will be highlighted in the *tab list* at the top of the display, which lists all the modules available in the patch. At any moment, you can go to another module by turning the *display encoder* D.

The module editor screen displays its parameters four by four, at the bottom of the display. You can manipulate them by **turning** the *parameter encoder* 1 2 3 4 corresponding to the one you're interested in.

You can also **zoom** on a specific parameter if you need to, by pressing its corresponding encoder for half a second.

To access the other rows of parameters for the current module, **press** the *matrix buttons* . The lights adjacent to them show how much rows there are, and what the currently selected row is.

You can leave the module editor screen and go back to the previous one by **pressing** .

Using the PC/Mac editor

For a more comfortable editing experience, you can find the PC/Mac editor *on our website*, alongside the latest update. Please always make sure you're using the editor that came with the update running on your Anyma Phi.

With your Anyma Phi freshly updated, you can now use the computer-based editor application that came with the update package.

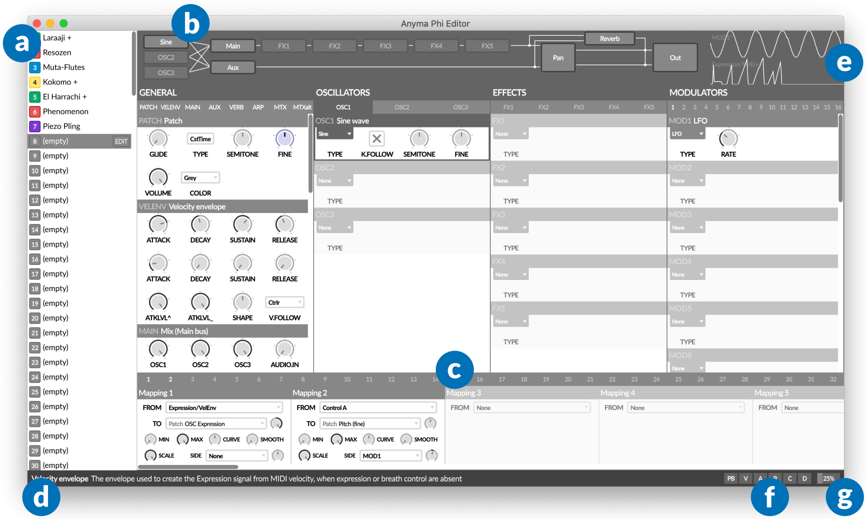
Connecting the editor

If this is your first time using the editor, please use the following instructions:

- Power on your Anyma Phi.
- Connect the *USB DEVICE* port of your Anyma Phi to your computer using a mini-USB cable.
- Launch the editor application.
- Select the entry named "Anyma Phi" in the *INPUT* and *OUTPUT* lists.
- Click the *CONNECT* button.

You should now be able to see all the patches in your Anyma Phi and edit them. Each time you launch the editor, it will try to connect to your Anyma Phi automatically. If it isn't connected to your computer, you will have to connect it and click the *CONNECT* button again to have access to your patches.

Using the editor



a Patch list

Displays the bank of your Anyma Phi.
Click on a patch to select and edit it.
After editing a patch, make sure you **save** it first using Ctrl/Cmd+S, 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+C and Ctrl/Cmd+V.
You can export the bank to a file using the relevant command in the *File* menu.

b **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.

c **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.

d **Contextual help**

Offers details about the element under the mouse pointer.

e **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.

f **External control meters**

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

g **Patch load meter**

Display the amount of resources used by the current patch.

The resources must stay under 100% to avoid glitches and dropouts.

Settings

Your **Anyma Phi** offers a *settings menu* where you can perform several commands, but also customize how your **Anyma Phi** reacts to MIDI, as well as various other settings such as the brightness of the front panel LEDs.

To enter the *settings menu*, press the *display encoder* D. You can then select a menu item by turning D, and confirm your choice using ▶. To go back to the previous screen, press ◀.

While in the matrix

Copy}} {{ Copy the current matrix (normal or alternate version, depending on which is active in the matrix screen) to the clipboard. }}

Paste
Paste the matrix in the clipboard, replacing the currently active version of the matrix (normal or alternate, depending on which is active).

Patch commands

Copy
Copy the current patch to the clipboard.

Paste
Paste the patch in the clipboard, replacing the current one, and adding "+" at the end of the patch name. However, the changes will not be automatically saved, allowing you to return to the previously saved version by switching to another patch and going back.

Save
Save the current patch.

This command is only available when the patch has unsaved modifications.

Reset
Reset the current patch to an empty one. However, the changes will not be automatically saved, allowing you to return to the previously saved version by switching to another patch and going back.

MIDI settings

MIDI channel
\aodyosettingsmark{Copy

\aodyosettingsmark{MIDI channel

Copy the current matrix (normal or alternate version, depending on which is active in the matrix screen) to the clipboard.

Paste
Paste the matrix in the clipboard, replacing the currently active version of the matrix (normal or alternate, depending on which is active).

Patch commands

Copy
Copy the current patch to the clipboard.

Paste
Paste the patch in the clipboard, replacing the current one, and adding "+" at the end of the patch name. However, the changes will not be automatically saved, allowing you to return to the previously saved version by switching to another patch and going back.

Save
Save the current patch.

This command is only available when the patch has unsaved modifications.

Reset
Reset the current patch to an empty one. However, the changes will not be automatically saved, allowing you to return to the previously saved version by switching to another patch and going back.

MIDI settings

MIDI channel} Set the MIDI channel that the Anyma Phi will respond to. You can use this to filter out messages from other controllers that go through your MIDI setup to another synth but should not be taken into account by your Anyma Phi.

```
**Default**: All
```

Bend range
Set the amount of semitones added or removed when pitch-bending at extreme values.

Default: ±2st

CC mapping

Expression CC, ...and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as the *expression* signal in the synth, for example by wind controllers. When either message is absent, the synth will use the velocity envelope of the patch to generate the expression signal.

Default: CC 2 (*Breath control*) and 11 (*Expression*)

Control A CC, ...and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control A* in the synth.

Default: CC 1 (*Modulation wheel*) and Off

Control B CC, ...and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control B* in the synth.

Default: CC 75 and 76

Control C CC, ...and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control C* in the synth.

Default: CC 76 and AT (*Channel aftertouch*)

Control D CC, ...and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control D* in the synth.

Default: CC 77 and Off

System settings

LED brightness
Make the front panel LEDs darker (1) or brighter (8).

Default: 4

Encoder accel.
Fine-tune encoder acceleration; i.e., how far you can change a value when quickly turning the encoder.

Default: 3

Volume calibr.
Enter the volume knob calibration screen. In this screen, you can modify raw *Minimum* and *Range* values that define how the raw input of the volume knob is used to calculate a volume in percents.

A basic calibration procedure would be as follows. First, turn down the volume knob to its physical minimum, and set the *Minimum* value on the calibration screen so that *Current* shows 1% or 2%, and increase *Minimum* a bit more until *Current* shows 0%. Then, turn up the volume knob to its physical maximum, and set the *Range* value on the calibration screen so that *Current* shows 98% or 99%, and decrease *Range* a bit more until *Current* shows 100%. You can then check that the volume knob is well calibrated by turning it over all its range and observing that *Current* goes from 0% to 100%, and exit the calibration screen by pressing ◀.

```
**Default**: _Minimum_ 156, _Range_ 93
```

{{SETTING: Diagnostics
Set the MIDI channel that the Anyma Phi will respond to. You can use this to filter out messages from other controllers that go through your MIDI setup to another synth but should not be taken into account by your Anyma Phi.

Default: All

Bend range
Set the amount of semitones added or removed when pitch-bending at extreme values.

Default: ±2st

CC mapping

Expression CC, …and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as the *expression* signal in the synth, for example by wind controllers. When either message is absent, the synth will use the velocity envelope of the patch to generate the expression signal.

Default: CC 2 (*Breath control*) and 11 (*Expression*)

Control A CC, …and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control A* in the synth.

Default: CC 1 (*Modulation wheel*) and Off

Control B CC, …and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control B* in the synth.

Default: CC 75 and 76

Control C CC, …and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control C* in the synth.

Default: CC 76 and AT (*Channel aftertouch*)

Control D CC, …and CC
Set up to two MIDI messages (*CC* or *Aftertouch*) used as *External control D* in the synth.

Default: CC 77 and Off

System settings

LED brightness

Make the front panel LEDs darker (1) or brighter (8).

Default: 4

Encoder accel.

Fine-tune encoder acceleration; i.e., how far you can change a value when quickly turning the encoder.

Default: 3

Volume calibr.

Enter the volume knob calibration screen. In this screen, you can modify raw *Minimum* and *Range* values that define how the raw input of the volume knob is used to calculate a volume in percents.

A basic calibration procedure would be as follows. First, turn down the volume knob to its physical minimum, and set the *Minimum* value on the calibration screen so that *Current* shows 1% or 2%, and increase *Minimum* a bit more until *Current* shows 0%. Then, turn up the volume knob to its physical maximum, and set the *Range* value on the calibration screen so that *Current* shows 98% or 99%, and decrease *Range* a bit more until *Current* shows 100%. You can then check that the volume knob is well calibrated by turning it over all its range and observing that *Current* goes from 0% to 100%, and exit the calibration screen by pressing ◀.

```
**Default**: _Minimum_ 156, _Range_ 93
```

{{SETTING: Diagnostics} Enter the *Diagnostics* screen, which you can use for troubleshooting with customer service.

Reset settings
Reset all the settings to their default values.

Shortcuts

Global actions

Press D
Enter the *settings menu*.
Press ⇧ + ▶
Save the current patch.
Turn 1 2 3 4
Modify a specific parameter (shown in the bottom part of the display).
Press 1 2 3 4
Select a specific parameter (shown in the bottom part of the display).
Keep 1 2 3 4 pressed for more than half a second
Zoom on a specific parameter (shown in the bottom part of the display).

MODULE NAVIGATION

The following actions allow you to select a specific module in the patch and edit it. They all require that ⇧ is held pressed during the entire time. You can combine any of these actions to make navigation easier. Once you've selected the module you're looking for, release ⇧ to confirm.

Turn ⇧ + D
Scroll the module list to select a particular one.
Press/hold ⇧ + ♨
Select the next module in the *General* section (*Patch*, *VelEnv*, *Main bus*, *Aux bus*, *Reverb*, etc.).

Press/hold +
Select the next oscillator (*OSC1* to *OSC3*).
Press/hold +
Select the next effect (*SFX1* to *SFX5*).
Press/hold +
Select the next modulator (*MOD1* to *MOD16*).
Press/hold +
Select the next mapping (*MAP1* to *MAP32*).

While in the matrix

Turn D
Select another patch, forgetting the current changes up until the last time it has been saved.
Press
Switch between the normal and the *alternate* version of the matrix.
Press
Select a specific line of the matrix.

In a *Related mappings* screen

Turn D
Select another related mapping in the tab list at the top of the display. The last tab allows you to create a new mapping if there's room.

In a *Module editor* screen

Turn D
Select another patch module from the tab list at the top of the display.

In the *settings menu*

The global shortcuts do not apply in the *settings menu*.

Turn D
Select a menu item. When editing a setting, select a value.
Press ◀
Return to the previous screen.
Press D or ▶
Confirm your selection.

On startup

The following shortcuts only apply on startup: keep holding the indicated buttons while powering on your Anyma Phi, and release them once the display shows your action has been taken into account.

Power on while holding ⇧
Enter the *update mode*, allowing you to transfer a new update into your Anyma Phi.
Power on while holding
Directly enter the *diagnostics screen*.
Power on while holding
Reset the settings to their default values without asking for confirmation.

Release notes

v0.8.90

- Added a pitch-bend range setting.
- The labels of the five effects in the audio signal chain are now *SFX1* to *SFX5* (previously *FX1* to *FX5*), so as to avoid confusion with the more generic *FX1* and *FX2* controls on the Global line of the matrix.
- On the Mac version of the editor, Ctrl+clicking now behaves like right-clicking.
- On the Windows version of the editor, importing/exporting of patches and banks through the menu now works under recent Windows versions.
- Scrolling through patches now displays the patch list, until scrolling stops and the patch is loaded.
- Saving an empty patch will change its name to *User#* (with # being the patch number).
- Added *Copy* and *Paste* commands for matrices and patches, as well as *Save* and *Reset* commands for patches, in the settings menu.
- When performing a command (like *Save*), a short notification now appears on the display.
- The titles of unused matrix controls (those not used in mappings) are greyed to better show which matrix controls are supported by

a patch. This doesn't prevent from changing their value or making new mappings.

- Added a volume knob calibration screen, accessible in the settings menu (*System* section). See the related entry in the *Settings* chapter of the user manual for more details and a manual calibration procedure.
- Patches with unsaved changes are indicated by a little diamond icon on the top right corner of the display, while in the matrix.
- Added new diagnostics for improved error reporting.
- Increased USB host compatibility with USB-MIDI devices in some very specific cases.
- By default, empty patches will now have Ctrl. A mapped to *Morph*, and Ctrl. B mapped to vibrato.
- Improved the resonators' behavior when no note has still been received.
- Fixed string display errors in some parts of the editor.

v0.8.80

- Added a *reset settings* startup shortcut.
- Improved navigation in the *settings menu*: pressing D now confirms the selection.
- Added an *Encoder acceleration* setting.

v0.8.70-5ea1262

Initial version shipped for super-early adopters.

Complementary information

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

Technical features

- Hybrid monophonic synthesizer
- Stereo audio outputs:
  - 1/4" mono line jack outputs
  - 1/4" stereo headphone jack output
- 1/4" stereo jack audio input (internal piezoelectric mic when external input unplugged)
- MIDI inputs and outputs via USB and DIN ports
- Powerful digital semi-modular synthesizer engine
  - Memory: 200 patches
  - 3 oscillator slots
  - 5 effect slots along 2 serial buses (Main, Aux) + 1 mono reverb
  - 16 modulator slots, including envelope generators, LFO, curves, interpolators, slew limiters, etc.
  - 32 mapping slots (virtual patch cords), each allowing to control any synth parameter (including another mapping) using a modulator or a controller input, with a sidechain input
  - Responsive to pitch, pitch-bend, velocity, breath control / expression, as well as 4 other controller inputs

Credits and licenses

Idea and design
Laurent Pouillard, Romain Bricout, Jonathan Aceituno, Ludovic Potier
Hardware design and development
Laurent Pouillard, Ludovic Potier
Sound design
Romain Bricout
Software design and development, user manual
Jonathan Aceituno
Marketing and social media
María Gómez
Sales
Matthias Couche
Testing and support
Maxence Fulconis

Portions of the synth use code adapted from *Mutable Instruments' Eurorack modules* made by Emilie Gillet.

> 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 synth 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.

Contact: Paul Mineiro `paul@mineiro.com`

Our internal software uses font *Silkscreen*, and the PC/Mac editor uses fonts *Lato* and *Font Awesome*, all licensed under the *SIL Open Font License*.

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 your **Anyma Phi**.

General

Module name	Label
Patch Global patch parameters	`PATCH`
Velocity envelope The envelope used to create the Expression signal from MIDI velocity, when expression or breath control are absent	`VELENV`
Mix (Main bus) The mixer for the Main bus	`MAIN`
Mix (Aux bus) The mixer for the Aux bus	`AUX`
Reverb The reverb effect at the end of the signal chain	`VERB`

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.

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) (DECAY)

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 Phi.

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 Phi.

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.
0 to 100%

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%

Oscillators

Module name	Label
Modal resonator Vibrating structure simulator	Modal
String resonator Vibrating string simulator	String
Windsyo Complex reed-based physical models	Windsyo
Artin PLUK Simple plucked string	PLUK
Artin BOWD Simple bowed string	BOWD
Artin BLOW Simple single-reed wind	BLOW
Artin FLUT Simple flute	FLUT
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
Artin SUB Waveform with sub-oscillator	SUB
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
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

Module name	Label
Artin PRTC Particle system simulator	PRTC
Artin QPSK Telecommunication data generator	QPSK

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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*

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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*

Artin PLUK (`PLUK`)

Simple plucked string.

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

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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%

Artin BOWD (`BOWD`)

Simple bowed string.

Simulates a bowed string using a physical model.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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 BELL (BELL)

Additive bell sound synthesizer.

Simulates a bell using a bank of decaying sine waves.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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%

Artin DRUM (`DRUM`)

Additive metal drum synthesizer.

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

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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%

Artin KICK (KICK)

808-style kick drum.

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

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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%

Artin CYMB (CYMB)

808-style cymbal.

Produces the typical cymbal sounds found on analog drum machines.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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%

Sine wave (`Sine`)

Pure tone without any harmonics.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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

Artin SUB (**SUB**)

Waveform with sub-oscillator.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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 BUZZ (BUZZ)

One to many sine waves.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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 (WAVEFORM)
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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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 (`FORMANT2`)
The frequency of the second formant.
-64 to +63 semitones

Third formant (`FORMANT3`)
The frequency of the third formant.
-64 to +63 semitones

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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%

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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%

White noise (`Noise`)

A simple white noise generator.

Artin NOIS (`NOIS`)

Filtered noise.

White noise filtered through a resonant multi-mode filter.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller’s pitch.

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.

Keyfollow (`K.FOLLOW`)
Whether the oscillator follows the controller's pitch.

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%

Artin CLOU (CLOU)

Granular cloud generator.

Produces random noise grains at various temporal intervals.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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.

Keyfollow (K.FOLLOW)
Whether the oscillator follows the controller's pitch.

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 (BAUDRATE)
0 to 100%

Data modulation (MOD)
0 to 100%

Effects

Module name	Label
Modal resonator Vibrating structure simulator	`Modal`
String resonator Vibrating string	`String`
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`
Amplifier A saturating amplifier	`AMP`
Overdrive Saturates without increasing volume	`Drive`
Cross-fader Balances between two inputs	`XFader`
Cross-fader with drive Cross-fader with drive controls	`XFadeD`
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`
Phaser Six-stage phase shifter	`Phaser`
Pitch-shifter Transposes the input	`PShiftr`
Delay Delay line with feedback and damping	`Delay`
Reverb Mono reverberation effect	`Reverb`

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.
0 to 100%

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

Keyfollow (K.FOLLOW)
Whether the effect follows the controller's pitch.

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.
0 to 100%

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

Keyfollow (`K.FOLLOW`)

Whether the effect follows the controller's pitch.

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

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.
0 to 100%

Filter mode (MODE)
The type of filter reponse.

- *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.
0 to 100%

Filter mode (`MODE`)
The type of filter reponse.

- *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.
0 to 100%

Low-pass (LOWPASS)
The low-pass filter frequency.
0 to 100%

High-pass (HIPASS)
The high-pass filter frequency.
0 to 100%

Peak frequency (PEAKFREQ)
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%

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.
0 to 100%

Gain (GAIN)
The gain applied to the input signal.
0 to 400%

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.
0 to 100%

Drive (`DRIVE`)
The level of overdrive applied to the input signal.
0 to 100%

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.
0 to 100%

Mix (`MIX`)
The amount of cross-fading after applying drive (this bus on the left, the other bus on the right).
0 to 100%

Cross-folder (`XFold`)

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.
0 to 100%

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%

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.
0 to 100%

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.

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.
0 to 100%

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%

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.
0 to 100%

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 (OPERATOR)
Selects the comparison operator to use.
0 to 100%

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.
0 to 100%

Delay time (TIME)
The time between the original and the delayed signal.
0 to 26.5 ms

Feedback (FEEDBACK)
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 6.50 ms

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.
0 to 100%

Rate (RATE)
The speed of phase shifting.
0.07 to 14.80 Hz

Feedback (FEEDBACK)
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.
0 to 100%

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

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.
0 to 100%

Delay time (`TIME`)
The time between the original and the delayed signal.
0 to 1000 ms

Feedback (`FEEDBACK`)
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.
0 to 100%

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%

Modulators

Module name	Label
Envelope Simple envelope generator	ENV
LFO Simple low-frequency oscillator	LFO
DAHDSR Envelope DAHDSR envelope generator	DAHDSR
Advanced LFO Advanced low-frequency oscillator with shape and fade-in controls	LFO+
Xform General-purpose value transform	XFORM
Clamp Limit a value to an interval	CLAMP
Wrap Wrap a value around an interval	WRAP
Fold Fold a value inside an interval	FOLD
Interpolate Cross-fade between two values	LERP
Interpolate (4-point) Interpolate between four points	LERP4
Calculate Perform successive operations on a series of values; e.g., (1+2)*3+4	CALC
Curve Apply a curve to a value	CURVE
Quantize Reduce the resolution of a value	QTZ
Smooth Smooth out the variations of a value	SMOOTH
Accumulate Accumulate a value or variations over time	ACCU
Impulse Generate an impulse from a value and a trigger	IMPULSE
Count Count occurrences of a trigger	COUNT
Time Measure the time since a trigger	TIME
Latch Capture a value when a trigger occurs	LATCH
Minimum Keep the minimum of a value since a trigger	MIN
Maximum Keep the maximum of a value since a trigger	MAX
Envelope follower Transforms an audio signal into a smoothed value	FOLLOW

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

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

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)
Whether the controller controls the gate.

Retrigger (RETRIG)
Whether to retrigger on a new note.

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.

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)
The 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 (SHAPEVAR)
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 (PHASE)

The amount of phase of the waveform.

0 to 100%

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

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.POS)
The position of point P1.
0 to 100%

P2 Position (P2.POS)
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., (1+2)*3+4.

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.

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

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.

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.

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

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.

Envelope follower (`FOLLOW`)

Transforms 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

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%

Aodyo SAS
11B avenue de l'Harmonie
59650 Villeneuve d'Ascq
France

contact@aodyo.com
www.aodyo.com

AODYO INSTRUMENTS